ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/386083346

Uncertain Automata and Uncertain Graph Grammar

Article · November 2024

CITATION
1

READS
14

2 authors:

Takaaki Fujita
124 PUBLICATIONS **298** CITATIONS

SEE PROFILE

Florentin Smarandache
University of New Mexico Gallup
4,048 PUBLICATIONS **52,338** CITATIONS

SEE PROFILE

All content following this page was uploaded by Takaaki Fujita on 24 November 2024.

The user has requested enhancement of the downloaded file.

Neutrosophic Sets and Systems, {Special Issue: Advances in SuperHyperStructures and Applied Neutrosophic Theories)}, Vol. 74, 2024

University of New Mexico

Uncertain Automata and Uncertain Graph Grammar

Takaaki Fujita [1], and Florentin Smarandache [2]

[1]∗ Independent Researcher, Shinjuku, Shinjuku-ku, Tokyo, Japan. t171d603@gunma-u.ac.jp

[2] University of New Mexico, Gallup Campus, NM 87301, USA. fsmarandache@gmail.com :

Abstract. Graph theory has been widely studied, resulting in numerous applications across various felds. Among its many topics, Automata and Graph Grammar have emerged as signifcant areas of research. This paper delves into these concepts, emphasizing their adaptation o uncertain frameworks like Fuzzy, Neutro- sophic, Vague, Turiyam Neutrosophic, and Plithogenic systems. By integrating uncertainty into traditional graph theoretical models, the paper aims to address ongoing research challenges and expand the scope of these models.

Keywords: Neutrosophic set, Fuzzy set, Graph Grammar, Automata

1. Introduction

1.1. *Graph Grammar and Automata*

Graph theory has been a subject of extensive research, leading to numerous applications across various domains [19,39–41,78,122,179]. Among its diverse topics, Graph Automata and Graph Grammar stand out as key areas of study within graph theory and its related concepts. This paper addresses Automata and Graph Grammar.

Automata are mathematical models representing abstract machines that process input sequences, transition between states, and generate outputs according to predefined rules [18,33,82,110,131,132]. Extensive research has been conducted on automata, resulting in various related concepts, including Finite Automata [35,124], Pushdown Automata [5,54,175,180], Infinite Automata [123, 177, 178], Linear-Bounded Automata [116], and Weighted Automata [42, 43, 113].

Graph Grammar is a formal system that defines transformation rules for modifying graph structures, modeling dynamic changes by adding, deleting, or replacing vertices and edges. It

has been extensively studied, with many related concepts explored [47,48,48,49,128]. Examples include Triple Graph Grammars [101,103,134,135], Context-Free Graph Grammars [6,53,121, 181], NLC Graph Grammars [46,57,84,85], Apex Graph Grammars [51,52,100], and Attributed Graph Grammars [75,129].

1.2. *Uncertain concepts*

Various concepts for handling uncertainty are continuously being researched to address the unpredictability of the world [7–11, 13, 149, 151, 152, 160, 165, 168, 170–172]. Among them, this paper focuses on fuzzy sets and neutrosophic sets. A fuzzy set assigns each element a membership degree between 0 and 1, representing partial membership [37, 44, 45, 87, 176, 188, 191, 192]. A neutrosophic set assigns three values—truth, indeterminacy, and falsity—to each element, with values ranging from 0 to 1 [14, 28, 50, 117, 149, 150, 172, 182]. A Turiyam Neutrosophic set assigns four values: truth, indeterminacy, falsity, and liberal state, all within the [0,1] range [69, 145–147]. These set concepts play significant roles in modeling uncertainty across various fields [61, 63, 64].

Concepts for handling uncertainty have been proposed not only in set theory but also in other fields. For example, in graph theory, various concepts have been explored, such as fuzzy graphs [115,127], neutrosophic graphs [12,15,29,60,80,83,90,130,159,169], vague graphs [27,126,133], Turiyam Neutrosophic graphs [69–71], and Plithogenic graphs [58, 102, 109]. Based on these points, research on Uncertain concepts is essential.

1.3. *Our contribution*

Based on the above, research on Graph Grammar and Automata is both significant and valuable. There remains considerable scope for further exploration, especially regarding uncertain concepts within Graph Grammar and Automata. Therefore, this paper will examine the concepts of Graph Grammar and Automata in the contexts of Fuzzy, Neutrosophic, Vague, Turiyam Neutrosophic, and Plithogenic frameworks.

This paper's structure is briefly outlined as follows. Section 2 examines Uncertain Automata, Section 3 explores Graph Grammar, and Section 4 discusses future perspectives.

2. **Automata**

Automata are mathematical models that represent computational machines or systems, processing input sequences to change states and produce outputs [18, 33, 82, 110, 131, 132]. In this section, we examine the concepts of Automata in the contexts of Fuzzy, Neutrosophic, Vague, Turiyam Neutrosophic, and Plithogenic frameworks.

2.1. *Fuzzy general finite Automata*

Fuzzy General Finite Automata are a fuzzy variant of General Finite Automata, widely explored in various studies [2,21,34]. In the context of automata theory and its application to fuzzy concepts, several other models have been developed, including Universal Fuzzy Automata [107], Fuzzy Rough Automata [119,186], LB-valued General Fuzzy Automata [4], BL-General Fuzzy Automata [1,3,139], Intuitionistic General Fuzzy Automata [142], Vector General Fuzzy Automata [140,141], Fuzzy Cellular Automata [16,23,73], Circular Fuzzy Cellular Automata [24], and Fuzzy Tree Automata [106]. The definitions and theorems, including related concepts, are provided below.

Definition 2.1. (cf. [82]) A *Finite Automaton (FA)* is defined as a 5-tuple:

$$A = (Q, \Sigma, \delta, q_0, F),$$

where:

- Q is a finite set of states.
- Σ is a finite alphabet of input symbols.
- $\delta : Q \times \Sigma \to Q$ is the state transition function, mapping a state and input symbol to a new state.
- $q_0 \in Q$ is the start state.
- $F \subseteq Q$ is the set of accepting (or final) states.

An FA processes an input string $w = a_1 a_2 \cdots a_n$ from Σ^* as follows:

(1) Begin in the start state q_0.
(2) For each symbol a_i in w, compute the new state $q_{i+1} = \delta(q_i, a_i)$.
(3) If the computation ends in a state $q \in F$, the automaton *accepts* w; otherwise, it *rejects* w.

Definition 2.2. [188,189] A *fuzzy set* τ in a non-empty universe Y is a mapping $\tau : Y \to [0, 1]$. A *fuzzy relation* on Y is a fuzzy subset δ in $Y \times Y$. If τ is a fuzzy set in Y and δ is a fuzzy relation on Y, then δ is called a *fuzzy relation on* τ if

$$\delta(y, z) \leq \min\{\tau(y), \tau(z)\} \quad \text{for all } y, z \in Y.$$

Example 2.3. Let $Y = \{1, 2, 3, 4, 5\}$ be a finite universe, and consider the following fuzzy set τ defined on Y:

$$\tau(y) = \begin{cases} 0.8 & \text{if } y = 1, \\ 0.6 & \text{if } y = 2, \\ 0.4 & \text{if } y = 3, \\ 0.2 & \text{if } y = 4, \\ 0.0 & \text{if } y = 5. \end{cases}$$

Now, consider a fuzzy relation δ defined on $Y \times Y$:

$$\delta(y, z) = \begin{cases} 0.5 & \text{if } (y, z) = (1, 2) \text{ or } (2, 1), \\ 0.4 & \text{if } (y, z) = (2, 3) \text{ or } (3, 2), \\ 0.2 & \text{if } (y, z) = (3, 4) \text{ or } (4, 3), \\ 0.0 & \text{otherwise.} \end{cases}$$

We can verify that δ is a fuzzy relation on τ by checking the condition:

$$\delta(y, z) \leq \min\{\tau(y), \tau(z)\}.$$

For instance:

- For $(y, z) = (1, 2)$, $\delta(1, 2) = 0.5$ and $\min\{\tau(1), \tau(2)\} = \min\{0.8, 0.6\} = 0.6$, so $0.5 \leq 0.6$.
- For $(y, z) = (2, 3)$, $\delta(2, 3) = 0.4$ and $\min\{\tau(2), \tau(3)\} = \min\{0.6, 0.4\} = 0.4$, so $0.4 \leq 0.4$.
- For $(y, z) = (3, 4)$, $\delta(3, 4) = 0.2$ and $\min\{\tau(3), \tau(4)\} = \min\{0.4, 0.2\} = 0.2$, so $0.2 \leq 0.2$.

Thus, δ is a valid fuzzy relation on τ.

Definition 2.4. [2, 21, 34] A *General Fuzzy Automaton (GFA)* is defined as an eight-tuple machine:

$$\tilde{F} = (Q, \Sigma, \tilde{R}, Z, \tilde{\delta}, \omega, F_1, F_2),$$

where:

- Q is a finite set of states, $Q = \{q_1, q_2, \ldots, q_n\}$.
- Σ is a finite set of input symbols, $\Sigma = \{a_1, a_2, \ldots, a_m\}$.
- $\tilde{R}$ is the set of fuzzy start states, $\tilde{R} \subseteq \tilde{P}(Q)$, where $\tilde{P}(Q)$ is the fuzzy power set of Q.
- Z is a finite set of output symbols, $Z = \{b_1, b_2, \ldots, b_k\}$.
- $\tilde{\delta} : (Q \times [0, 1]) \times \Sigma \times Q \xrightarrow{F_1(\mu,\delta)} [0, 1]$ is the *augmented fuzzy transition function*, where μ is the membership value of a predecessor state, and δ is the transition weight.

- $\omega : Q \to Z$ is the non-fuzzy output function.
- $F_1 : [0,1] \times [0,1] \to [0,1]$ is the *membership assignment function*, with common forms such as $F_1(\mu, \delta) = \max(\mu, \delta)$, $\min(\mu, \delta)$, or $\frac{\mu+\delta}{2}$.
- $F_2 : [0,1]^* \to [0,1]$ is the *multi-membership resolution function*, used to resolve simultaneous transitions to the same state.

Definition 2.5. The membership value of the state q_j at time $t+1$ is given by:

$$\mu_{t+1}(q_j) = \tilde{\delta}((q_i, \mu_t(q_i)), a_k, q_j) = F_1(\mu_t(q_i), \delta(q_i, a_k, q_j)),$$

where:

- $\mu_t(q_i)$ is the membership value of state q_i at time t.
- $\delta(q_i, a_k, q_j)$ is the transition weight from q_i to q_j on input a_k.

Definition 2.6. If there are multiple transitions to the active state q_j, the membership values v_i from different transitions are combined using F_2:

$$\mu_{t+1}(q_j) = F_2\left(\{v_i\}_{i=1}^n\right) = F_2\left(\{F_1(\mu_t(q_i), \delta(q_i, a_k, q_j))\}_{i=1}^n\right),$$

where:

- n is the number of simultaneous transitions to state q_j at time $t+1$.

Proposition 2.7. *A General Fuzzy Automaton (GFA) is a generalization of a Finite Automaton (FA).*

Proof. Let $A = (Q, \Sigma, \delta, q_0, F)$ be a finite automaton. Define a GFA $\tilde{F}$ as follows:

- Use the same set of states: $Q_{\text{GFA}} = Q_{\text{FA}}$.
- Use the same alphabet: $\Sigma_{\text{GFA}} = \Sigma_{\text{FA}}$.
- Set the fuzzy start state: $\tilde{R} = \{q_0 \mapsto 1, q \mapsto 0 \text{ for all } q \neq q_0\}$.
- Define the fuzzy transition function:

$$\tilde{\delta}((q,1), a, q') = \begin{cases} 1 & \text{if } \delta(q,a) = q', \\ 0 & \text{otherwise.} \end{cases}$$

- Use a trivial output function: $\omega(q) = b_0$ for some fixed $b_0 \in Z$.
- Define $F_1(\mu, \delta) = \delta$, treating δ as a deterministic transition.
- Use F_2 as the identity function to avoid multi-membership conflicts.

With these definitions:

(1) Any deterministic transition in A is encoded as a transition with membership degree 1 in $\tilde{F}$.

(2) The computation of $\tilde{F}$ simulates the state changes of A exactly, as F_1 and F_2 are designed to preserve deterministic behavior.

Thus, any FA is a special case of a GFA where the fuzzy parameters are restricted to deterministic values. Therefore, $\tilde{F}$ generalizes A. □

2.2. *Neutrosophic general finite automata*

Neutrosophic General Finite Automata represent a Neutrosophic adaptation of General Finite Automata, extensively studied in various works [99]. Within the scope of automata theory and its application to Neutrosophic concepts, several models have emerged, including Composite Neutrosophic Finite Automata [98], Reverse Neutrosophic Automata [97], Single-Valued Neutrosophic Automata [94, 95, 112], and Interval Neutrosophic Automata [91–93, 96, 97]. The definitions and theorems, including related concepts, are provided below.

Definition 2.8. [149] Let X be a given set. A Neutrosophic Set A on X is characterized by three membership functions:

$$T_A : X \to [0,1], \quad I_A : X \to [0,1], \quad F_A : X \to [0,1],$$

where for each $x \in X$, the values $T_A(x)$, $I_A(x)$, and $F_A(x)$ represent the degree of truth, indeterminacy, and falsity, respectively. These values satisfy the following condition:

$$0 \leq T_A(x) + I_A(x) + F_A(x) \leq 3.$$

Definition 2.9. (cf. [94, 95, 99, 112]) A *General Neutrosophic Automaton (GNA)* is defined as an eight-tuple machine:

$$\tilde{F} = (Q, \Sigma, \tilde{R}, Z, \tilde{\delta}, \omega, F_1, F_2),$$

where:

- Q, Σ, Z, and ω are defined as in the GFA.
- $\tilde{R} = \{(q, \mu_1^0(q), \mu_2^0(q), \mu_3^0(q)) \mid q \in R\}$ is the set of *neutrosophic start states*, where:
  - $\mu_1^0(q)$ is the initial truth-membership value of state q,
  - $\mu_2^0(q)$ is the initial indeterminacy-membership value of state q,
  - $\mu_3^0(q)$ is the initial falsity-membership value of state q.
- $\tilde{\delta} : (Q \times [0,1] \times [0,1] \times [0,1]) \times \Sigma \times Q \xrightarrow{F_1(\mu,\delta)} [0,1] \times [0,1] \times [0,1]$ is the *neutrosophic augmented transition function*, where $\mu = (\mu_1, \mu_2, \mu_3)$ and $\delta = (\delta_1, \delta_2, \delta_3)$ represent the truth, indeterminacy, and falsity components.
- $F_1 = (F_1^{\wedge}, F_1^{\wedge\vee}, F_1^{\vee})$, where:
  - $F_1^{\wedge}(\mu_1, \delta_1)$ represents the truth-membership assignment,

  - $F_1^{\wedge\vee}(\mu_2, \delta_2)$ represents the indeterminacy-membership assignment,
  - $F_1^{\vee}(\mu_3, \delta_3)$ represents the falsity-membership assignment.
- $F_2 = (F_2^{\wedge}, F_2^{\wedge\vee}, F_2^{\vee})$ is the *multi-membership resolution function*, resolving truth, indeterminacy, and falsity components.

Definition 2.10. The neutrosophic membership values of state q_j at time $t+1$ are calculated as:

$$\begin{aligned}
\mu_{1,t+1}(q_j) &= \tilde{\delta}_1((q_i, \mu_{1,t}(q_i)), a_k, q_j) = F_1^{\wedge}(\mu_{1,t}(q_i), \delta_1(q_i, a_k, q_j)),\\
\mu_{2,t+1}(q_j) &= \tilde{\delta}_2((q_i, \mu_{2,t}(q_i)), a_k, q_j) = F_1^{\wedge\vee}(\mu_{2,t}(q_i), \delta_2(q_i, a_k, q_j)),\\
\mu_{3,t+1}(q_j) &= \tilde{\delta}_3((q_i, \mu_{3,t}(q_i)), a_k, q_j) = F_1^{\vee}(\mu_{3,t}(q_i), \delta_3(q_i, a_k, q_j)),
\end{aligned}$$

where:

- $\mu_{1,t}(q_i)$, $\mu_{2,t}(q_i)$, and $\mu_{3,t}(q_i)$ represent the truth, indeterminacy, and falsity membership values of state q_i at time t.

Definition 2.11. The simultaneous transitions to the same state q_j are resolved by F_2:

$$\begin{aligned}
\mu_{1,t+1}(q_j) &= F_2^{\wedge}\left(\{F_1^{\wedge}(\mu_{1,t}(q_i), \delta_1(q_i, a_k, q_j))\}_{i=1}^{n}\right),\\
\mu_{2,t+1}(q_j) &= F_2^{\wedge\vee}\left(\{F_1^{\wedge\vee}(\mu_{2,t}(q_i), \delta_2(q_i, a_k, q_j))\}_{i=1}^{n}\right),\\
\mu_{3,t+1}(q_j) &= F_2^{\vee}\left(\{F_1^{\vee}(\mu_{3,t}(q_i), \delta_3(q_i, a_k, q_j))\}_{i=1}^{n}\right).
\end{aligned}$$

Proposition 2.12. *A General Neutrosophic Automaton (GNA) is a generalization of a General Fuzzy Automaton (GFA).*

Proof. Let $\tilde{F}_{\mathrm{GFA}} = (Q, \Sigma, \tilde{R}, Z, \tilde{\delta}, \omega, F_1, F_2)$ be a General Fuzzy Automaton. Define a GNA $\tilde{F}_{\mathrm{GNA}} = (Q, \Sigma, \tilde{R}_{\mathrm{N}}, Z, \tilde{\delta}_{\mathrm{N}}, \omega, F_1^{\mathrm{N}}, F_2^{\mathrm{N}})$ as follows:

- Use the same set of states, input symbols, and output symbols: $Q_{\mathrm{GNA}} = Q_{\mathrm{GFA}}$, $\Sigma_{\mathrm{GNA}} = \Sigma_{\mathrm{GFA}}$, $Z_{\mathrm{GNA}} = Z_{\mathrm{GFA}}$.
- Define the neutrosophic start states:

$$\tilde{R}_{\mathrm{N}} = \{(q, \mu_1, \mu_2, \mu_3) \mid q \in Q, \mu_1 = \mu, \mu_2 = 0, \mu_3 = 1 - \mu \text{ for } \mu \in \tilde{R}\}.$$

- Extend the fuzzy transition function $\tilde{\delta}$ to the neutrosophic transition function $\tilde{\delta}_{\mathrm{N}}$:

$$\tilde{\delta}_{\mathrm{N}}((q, \mu_1, \mu_2, \mu_3), a, q') = \begin{cases} (\tilde{\delta}(q, a, q'), 0, 1 - \tilde{\delta}(q, a, q')) & \text{if } \tilde{\delta}(q, a, q') \neq 0,\\ (0, 0, 1) & \text{otherwise.} \end{cases}$$

- Define F_1^{N} to operate component-wise:

$$F_1^{\mathrm{N}}((\mu_1, \mu_2, \mu_3), (\delta_1, \delta_2, \delta_3)) = (F_1(\mu_1, \delta_1), 0, 1 - F_1(\mu_1, \delta_1)).$$

- Use F_2^{N} as a component-wise extension of F_2:

$$F_2^{\mathrm{N}}(\{(\mu_{1,i}, \mu_{2,i}, \mu_{3,i})\}) = (F_2(\{\mu_{1,i}\}), 0, 1 - F_2(\{\mu_{1,i}\})).$$

By construction, $\tilde{F}_{\mathrm{GNA}}$ reproduces the behavior of $\tilde{F}_{\mathrm{GFA}}$ when restricted to the fuzzy case, where indeterminacy is always 0 and falsity is the complement of truth. Thus, GNA generalizes GFA. □

Proposition 2.13. *A General Neutrosophic Automaton (GNA) is a generalization of a Finite Automaton (FA).*

Proof. Let $A = (Q, \Sigma, \delta, q_0, F)$ be a finite automaton. Define a GNA $\tilde{F}_{\mathrm{GNA}}$ as follows:

- Use the same set of states and input symbols: $Q_{\mathrm{GNA}} = Q_{\mathrm{FA}}$, $\Sigma_{\mathrm{GNA}} = \Sigma_{\mathrm{FA}}$.
- Define the neutrosophic start state:

$$\tilde{R}_{\mathrm{N}} = \{(q, 1, 0, 0) \text{ if } q = q_0, (q, 0, 0, 1) \text{ otherwise.}\}.$$

- Define the neutrosophic transition function:

$$\tilde{\delta}_{\mathrm{N}}((q, \mu_1, \mu_2, \mu_3), a, q') = \begin{cases} (1, 0, 0) & \text{if } \delta(q, a) = q', \\ (0, 0, 1) & \text{otherwise.} \end{cases}$$

- Use trivial membership assignment and resolution functions:

$$F_1^{\mathrm{N}}((\mu_1, \mu_2, \mu_3), (\delta_1, \delta_2, \delta_3)) = (\delta_1, 0, 1 - \delta_1),$$

$$F_2^{\mathrm{N}}(\{(\mu_{1,i}, \mu_{2,i}, \mu_{3,i})\}) = (1, 0, 0).$$

Under this construction:

(1) Each state in A maps to a neutrosophic state in $\tilde{F}_{\mathrm{GNA}}$, with truth membership 1 and falsity membership 0 when active.
(2) The deterministic transitions of A are captured by the neutrosophic transition function with truth membership 1.

Thus, $\tilde{F}_{\mathrm{GNA}}$ reproduces the behavior of A as a special case where indeterminacy is always 0, truth is binary, and falsity is the complement of truth. Therefore, GNA generalizes FA. □

2.3. *Turiyam Neutrosophic general finite automata*

Turiyam Neutrosophic general finite automata represent a Turiyam Neutrosophic adaptation of General Finite Automata. The definitions and theorems, including related concepts, are provided below.

Definition 2.14. [69, 145] For Turiyam Neutrosophic sets μ and ν on E:

- *Order Relation:* $\mu \leq \nu$ if for all $e \in E$:

$$T_\mu(e) \leq T_\nu(e), \quad I_\mu(e) \geq I_\nu(e), \quad F_\mu(e) \geq F_\nu(e), \quad L_\mu(e) \geq L_\nu(e).$$

- *Maximum (Join):*

$$(\mu \vee \nu)(e) = (\max\{T_\mu(e), T_\nu(e)\},\ \min\{I_\mu(e), I_\nu(e)\},\ \min\{F_\mu(e), F_\nu(e)\},\ \min\{L_\mu(e), L_\nu(e)\})\,.$$

- *Minimum (Meet):*

$$(\mu \wedge \nu)(e) = (\min\{T_\mu(e), T_\nu(e)\},\ \max\{I_\mu(e), I_\nu(e)\},\ \max\{F_\mu(e), F_\nu(e)\},\ \max\{L_\mu(e), L_\nu(e)\})\,.$$

- *Support of μ:*

$$\mathrm{supp}(\mu) = \{e \in E : T_\mu(e) > 0\}.$$

- *Minimal Truth-Membership Value:*

$$m_T(\mu) = \min\{T_\mu(e) : e \in \mathrm{supp}(\mu)\}.$$

Definition 2.15. A *Turiyam Neutrosophic General Finite Automaton (TGFA)* is an eight-tuple machine defined as:

$$\tilde{F} = (Q, \Sigma, \tilde{R}, Z, \tilde{\delta}, \omega, F_1, F_2),$$

where:

(1) *States*:
- Q is a finite set of states:

$$Q = \{q_1, q_2, \ldots, q_n\}.$$

(2) *Input Symbols*:
- Σ is a finite set of input symbols:

$$\Sigma = \{a_1, a_2, \ldots, a_m\}.$$

(3) *Turiyam Neutrosophic Start States*:
- $\tilde{R}$ is the set of Turiyam Neutrosophic start states:

$$\tilde{R} = \left\{ \left(q, \mu_1^0(q), \mu_2^0(q), \mu_3^0(q), \mu_4^0(q)\right) \,\middle|\, q \in R \right\},$$

where:
  - $R \subseteq Q$ is the set of initial states.

- $\mu_1^0(q)$ is the initial truth-membership value of state q.
- $\mu_2^0(q)$ is the initial indeterminacy-membership value of state q.
- $\mu_3^0(q)$ is the initial falsity-membership value of state q.
- $\mu_4^0(q)$ is the initial liberal state-membership value of state q.

(4) *Output Symbols*:

- Z is a finite set of output symbols:

$$Z = \{b_1, b_2, \ldots, b_k\}.$$

(5) *Turiyam Neutrosophic Transition Function*:

- $\tilde{\delta} : (Q \times [0,1]^4) \times \Sigma \times Q \to [0,1]^4$ is the augmented Turiyam Neutrosophic transition function, defined by:

$$\tilde{\delta}\left((q_i, \mu_t(q_i)), a_k, q_j\right) = \left(\tilde{\delta}_1, \tilde{\delta}_2, \tilde{\delta}_3, \tilde{\delta}_4\right),$$

where:

- $\mu_t(q_i) = (\mu_{1,t}(q_i), \mu_{2,t}(q_i), \mu_{3,t}(q_i), \mu_{4,t}(q_i))$ represents the Turiyam Neutrosophic membership values of state q_i at time t.
- $\delta(q_i, a_k, q_j) = (\delta_1(q_i, a_k, q_j), \delta_2(q_i, a_k, q_j), \delta_3(q_i, a_k, q_j), \delta_4(q_i, a_k, q_j))$ are the Turiyam Neutrosophic transition weights.
- The components are computed using the membership assignment functions:

$$\begin{aligned}
\tilde{\delta}_1 &= F_1^{\wedge}\left(\mu_{1,t}(q_i), \delta_1(q_i, a_k, q_j)\right), \\
\tilde{\delta}_2 &= F_1^{\wedge\vee}\left(\mu_{2,t}(q_i), \delta_2(q_i, a_k, q_j)\right), \\
\tilde{\delta}_3 &= F_1^{\vee}\left(\mu_{3,t}(q_i), \delta_3(q_i, a_k, q_j)\right), \\
\tilde{\delta}_4 &= F_1^{\mathrm{L}}\left(\mu_{4,t}(q_i), \delta_4(q_i, a_k, q_j)\right).
\end{aligned}$$

(6) *Output Function*:

- $\omega : (Q \times [0,1]^4) \to Z$ is the non-fuzzy output function.

(7) *Membership Assignment Functions*:

- $F_1 = \left(F_1^{\wedge}, F_1^{\wedge\vee}, F_1^{\vee}, F_1^{\mathrm{L}}\right)$, where:
  - $F_1^{\wedge} : [0,1] \times [0,1] \to [0,1]$ assigns the truth-membership value.
  - $F_1^{\wedge\vee} : [0,1] \times [0,1] \to [0,1]$ assigns the indeterminacy-membership value.
  - $F_1^{\vee} : [0,1] \times [0,1] \to [0,1]$ assigns the falsity-membership value.
  - $F_1^{\mathrm{L}} : [0,1] \times [0,1] \to [0,1]$ assigns the liberal state-membership value.

(8) *Multi-Membership Resolution Functions*:

- $F_2 = \left(F_2^{\wedge}, F_2^{\wedge\vee}, F_2^{\vee}, F_2^{\mathrm{L}}\right)$, where:
  - Each F_2 component resolves multiple membership values when there are simultaneous transitions to the same state.

Definition 2.16. The Turiyam Neutrosophic membership values of state q_j at time $t+1$ are calculated as:

$$\begin{aligned}
\mu_{1,t+1}(q_j) &= \tilde{\delta}_1\left((q_i, \mu_{1,t}(q_i)), a_k, q_j\right) = F_1^{\wedge}\left(\mu_{1,t}(q_i), \delta_1(q_i, a_k, q_j)\right), \\
\mu_{2,t+1}(q_j) &= \tilde{\delta}_2\left((q_i, \mu_{2,t}(q_i)), a_k, q_j\right) = F_1^{\wedge\vee}\left(\mu_{2,t}(q_i), \delta_2(q_i, a_k, q_j)\right), \\
\mu_{3,t+1}(q_j) &= \tilde{\delta}_3\left((q_i, \mu_{3,t}(q_i)), a_k, q_j\right) = F_1^{\vee}\left(\mu_{3,t}(q_i), \delta_3(q_i, a_k, q_j)\right), \\
\mu_{4,t+1}(q_j) &= \tilde{\delta}_4\left((q_i, \mu_{4,t}(q_i)), a_k, q_j\right) = F_1^{\mathrm{L}}\left(\mu_{4,t}(q_i), \delta_4(q_i, a_k, q_j)\right).
\end{aligned}$$

Definition 2.17. If there are multiple transitions to the same state q_j, the membership values are resolved using F_2:

$$\begin{aligned}
\mu_{1,t+1}(q_j) &= F_2^{\wedge}\left(\left\{F_1^{\wedge}\left(\mu_{1,t}(q_i), \delta_1(q_i, a_k, q_j)\right)\right\}_{i=1}^{n}\right), \\
\mu_{2,t+1}(q_j) &= F_2^{\wedge\vee}\left(\left\{F_1^{\wedge\vee}\left(\mu_{2,t}(q_i), \delta_2(q_i, a_k, q_j)\right)\right\}_{i=1}^{n}\right), \\
\mu_{3,t+1}(q_j) &= F_2^{\vee}\left(\left\{F_1^{\vee}\left(\mu_{3,t}(q_i), \delta_3(q_i, a_k, q_j)\right)\right\}_{i=1}^{n}\right), \\
\mu_{4,t+1}(q_j) &= F_2^{\mathrm{L}}\left(\left\{F_1^{\mathrm{L}}\left(\mu_{4,t}(q_i), \delta_4(q_i, a_k, q_j)\right)\right\}_{i=1}^{n}\right),
\end{aligned}$$

where:

- n is the number of simultaneous transitions to state q_j at time $t+1$.
- $q_i \in Q$ are the predecessor states.

Theorem 2.18. *A Turiyam Neutrosophic General Finite Automaton (TGFA) can be transformed into a Fuzzy General Finite Automaton (GFA) or a Neutrosophic General Finite Automaton (GNA) by appropriately redefining its transition and membership functions.*

Proof. Let $\tilde{F}^T$ be a Turiyam Neutrosophic General Finite Automaton defined as:

$$\tilde{F}^T = (Q, \Sigma, \tilde{R}, Z, \tilde{\delta}^T, \omega, F_1^T, F_2^T),$$

where:

- $\tilde{\delta}^T : (Q \times [0,1]^4) \times \Sigma \times Q \to [0,1]^4$ is the augmented Turiyam Neutrosophic transition function.
- F_1^T and F_2^T are the membership assignment and resolution functions, respectively, with four components: truth, indeterminacy, falsity, and liberal state.

We consider Transformation to a Fuzzy General Finite Automaton. A Fuzzy General Finite Automaton $\tilde{F}^F$ is defined as:

$$\tilde{F}^F = (Q, \Sigma, \tilde{R}^F, Z, \tilde{\delta}^F, \omega, F_1^F, F_2^F),$$

where:

- The fuzzy start states $\tilde{R}^F$ are obtained from $\tilde{R}$ in the TGFA by retaining only the truth-membership component:

$$\tilde{R}^F = \{(q, \mu_1^0(q)) \mid (q, \mu_1^0(q), \mu_2^0(q), \mu_3^0(q), \mu_4^0(q)) \in \tilde{R}\}.$$

- The fuzzy transition function $\tilde{\delta}^F$ is derived from the truth component of $\tilde{\delta}^T$:

$$\tilde{\delta}^F\left((q_i, \mu_{1,t}(q_i)), a_k, q_j\right) = \tilde{\delta}_1\left((q_i, \mu_{1,t}(q_i)), a_k, q_j\right).$$

- The membership assignment function F_1^F corresponds to the truth-assignment component $F_1^\wedge$ of the TGFA:

$$F_1^F(\mu, \delta) = F_1^\wedge(\mu, \delta).$$

- The multi-membership resolution function F_2^F is defined as:

$$F_2^F\left(\{v_i\}\right) = F_2^\wedge\left(\{v_i\}\right),$$

  where v_i are the truth-membership values from simultaneous transitions.

Thus, the TGFA $\tilde{F}^T$ reduces to a GFA $\tilde{F}^F$, retaining only the truth-membership component and the corresponding fuzzy transition and resolution functions.

Next, we consider Transformation to a Neutrosophic General Finite Automaton. A Neutrosophic General Finite Automaton $\tilde{F}^N$ is defined as:

$$\tilde{F}^N = (Q, \Sigma, \tilde{R}^N, Z, \tilde{\delta}^N, \omega, F_1^N, F_2^N),$$

where:

- The neutrosophic start states $\tilde{R}^N$ are obtained from $\tilde{R}$ in the TGFA by retaining the truth, indeterminacy, and falsity components:

$$\tilde{R}^N = \{(q, \mu_1^0(q), \mu_2^0(q), \mu_3^0(q)) \mid (q, \mu_1^0(q), \mu_2^0(q), \mu_3^0(q), \mu_4^0(q)) \in \tilde{R}\}.$$

- The neutrosophic transition function $\tilde{\delta}^N$ is derived from the first three components of $\tilde{\delta}^T$:

$$\tilde{\delta}^N\left((q_i, \mu_t(q_i)), a_k, q_j\right) = \left(\tilde{\delta}_1, \tilde{\delta}_2, \tilde{\delta}_3\right).$$

- The membership assignment function F_1^N is defined by the truth, indeterminacy, and falsity components of the TGFA:

$$F_1^N = \left(F_1^\wedge, F_1^{\wedge\vee}, F_1^\vee\right).$$

- The multi-membership resolution function F_2^N is defined as:

$$F_2^N = \left(F_2^\wedge, F_2^{\wedge\vee}, F_2^\vee\right).$$

Thus, the TGFA $\tilde{F}^T$ reduces to a GNA $\tilde{F}^N$, retaining the truth, indeterminacy, and falsity components, along with their corresponding neutrosophic transition and resolution functions.

By appropriately redefining the start states, transition functions, membership assignment, and resolution functions, a Turiyam Neutrosophic General Finite Automaton (TGFA) can be transformed into either a Fuzzy General Finite Automaton (GFA) or a Neutrosophic General Finite Automaton (GNA). □

2.4. *Vague General Finite Automata*

Vague general finite automata represent a Vague adaptation of General Finite Automata. The definitions and theorems, including related concepts, are provided below.

Definition 2.19. (cf. [30, 32, 56, 72, 111, 125, 184]) A *Vague Set (VS) A* on a set X is defined as:

$$A = (t_A, f_A),$$

where:

- $t_A : X \to [0, 1]$ is the *truth-membership function.*
- $f_A : X \to [0, 1]$ is the *falsity-membership function.*

For any element $x \in X$, the following condition holds:

$$0 \leq t_A(x) + f_A(x) \leq 1.$$

Definition 2.20. A *Vague General Finite Automaton (VGFA)* is defined as an eight-tuple:

$$\tilde{F} = (Q, \Sigma, \tilde{R}, Z, \tilde{\delta}, \omega, F_1, F_2),$$

where:

(1) *States*:
- Q is a finite set of states:

$$Q = \{q_1, q_2, \ldots, q_n\}.$$

(2) *Input Symbols*:
- Σ is a finite set of input symbols:

$$\Sigma = \{a_1, a_2, \ldots, a_m\}.$$

(3) *Vague Start States*:
- $\tilde{R}$ is the set of vague start states:

$$\tilde{R} = \left\{ \left(q, \mu_t^0(q), \mu_f^0(q)\right) \,\middle|\, q \in R \right\},$$

where:

- $R \subseteq Q$ is the set of initial states.
- $\mu_t^0(q)$ is the initial truth-membership value of state q.
- $\mu_f^0(q)$ is the initial falsity-membership value of state q.
- For all $q \in R$:

$$0 \leq \mu_t^0(q) + \mu_f^0(q) \leq 1.$$

(4) *Output Symbols*:

- Z is a finite set of output symbols:

$$Z = \{b_1, b_2, \ldots, b_k\}.$$

(5) *Vague Transition Function*:

- $\tilde{\delta} : (Q \times [0,1]^2) \times \Sigma \times Q \to [0,1]^2$ is the augmented vague transition function, defined by:

$$\tilde{\delta}\left((q_i, \mu_t(q_i), \mu_f(q_i)), a_k, q_j\right) = \left(\tilde{\delta}_t, \tilde{\delta}_f\right),$$

where:

- $\mu_t(q_i)$ and $\mu_f(q_i)$ are the truth and falsity membership values of state q_i.
- $\delta(q_i, a_k, q_j) = (\delta_t(q_i, a_k, q_j), \delta_f(q_i, a_k, q_j))$ are the transition weights.
- The components are computed using the membership assignment function F_1:

$$\tilde{\delta}_t = F_1^{\wedge}\left(\mu_t(q_i), \delta_t(q_i, a_k, q_j)\right),$$
$$\tilde{\delta}_f = F_1^{\vee}\left(\mu_f(q_i), \delta_f(q_i, a_k, q_j)\right).$$

(6) *Output Function*:

- $\omega : Q \to Z$ is the output function.

(7) *Membership Assignment Functions*:

- $F_1 = (F_1^{\wedge}, F_1^{\vee})$, where:
  - $F_1^{\wedge} : [0,1] \times [0,1] \to [0,1]$ assigns the truth-membership value.
  - $F_1^{\vee} : [0,1] \times [0,1] \to [0,1]$ assigns the falsity-membership value.
  - Common forms for $F_1^{\wedge}$ and $F_1^{\vee}$ include:

$$F_1^{\wedge}(\mu_t, \delta_t) = \min(\mu_t, \delta_t), \quad F_1^{\vee}(\mu_f, \delta_f) = \max(\mu_f, \delta_f).$$

(8) *Multi-Membership Resolution Function*:

- $F_2 = (F_2^{\wedge}, F_2^{\vee})$, where:
  - $F_2^{\wedge} : [0,1]^n \to [0,1]$ resolves multiple truth-membership values.
  - $F_2^{\vee} : [0,1]^n \to [0,1]$ resolves multiple falsity-membership values.
  - Common forms include:

$$F_2^{\wedge}\left(\{\mu_t^i\}_{i=1}^n\right) = \max_i \mu_t^i, \quad F_2^{\vee}\left(\{\mu_f^i\}_{i=1}^n\right) = \min_i \mu_f^i.$$

The membership values of the state q_j at time $t+1$ are calculated as:

$$\mu_{t+1}^{t}(q_j) = F_2^{\wedge}\left(\left\{F_1^{\wedge}\left(\mu_t^t(q_i), \delta_t(q_i, a_k, q_j)\right)\right\}_{i=1}^{n}\right),$$
$$\mu_{t+1}^{f}(q_j) = F_2^{\vee}\left(\left\{F_1^{\vee}\left(\mu_t^f(q_i), \delta_f(q_i, a_k, q_j)\right)\right\}_{i=1}^{n}\right),$$

where:

- n is the number of transitions leading to q_j at time $t+1$.
- $\mu_t^t(q_i)$ and $\mu_t^f(q_i)$ are the truth and falsity membership values of state q_i at time t.
- $\delta_t(q_i, a_k, q_j)$ and $\delta_f(q_i, a_k, q_j)$ are the transition truth and falsity weights.

Theorem 2.21. *A Neutrosophic General Finite Automaton (GNA) can be transformed into a Vague General Finite Automaton (VGFA) by appropriately redefining its transition and membership functions.*

Proof. Let $\tilde{F}^N$ be a Neutrosophic General Finite Automaton defined as:

$$\tilde{F}^N = (Q, \Sigma, \tilde{R}, Z, \tilde{\delta}^N, \omega, F_1^N, F_2^N),$$

where:

- $\tilde{\delta}^N : (Q \times [0,1]^3) \times \Sigma \times Q \to [0,1]^3$ is the neutrosophic transition function, which includes truth, indeterminacy, and falsity components.
- F_1^N and F_2^N are the membership assignment and resolution functions, respectively, with three components: truth, indeterminacy, and falsity.

A Vague General Finite Automaton $\tilde{F}^V$ is defined as:

$$\tilde{F}^V = (Q, \Sigma, \tilde{R}^V, Z, \tilde{\delta}^V, \omega, F_1^V, F_2^V),$$

where:

- *Vague Start States:* The vague start states $\tilde{R}^V$ are obtained from the neutrosophic start states $\tilde{R}^N$ by combining the truth and falsity components, while ignoring the indeterminacy component:

  $$\tilde{R}^V = \{(q, \mu_t^0(q), \mu_f^0(q)) \mid (q, \mu_1^0(q), \mu_2^0(q), \mu_3^0(q)) \in \tilde{R}^N\},$$

  where:

  $$\mu_t^0(q) = \mu_1^0(q),$$
  $$\mu_f^0(q) = \mu_3^0(q).$$

- *Vague Transition Function:* The vague transition function $\tilde{\delta}^V$ is derived from the neutrosophic transition function $\tilde{\delta}^N$ by retaining only the truth and falsity components:

  $$\tilde{\delta}^V\left((q_i, \mu_t(q_i), \mu_f(q_i)), a_k, q_j\right) = \left(\tilde{\delta}_t, \tilde{\delta}_f\right),$$

where:

$$\tilde{\delta}_t = \tilde{\delta}_1\left((q_i, \mu_t(q_i)), a_k, q_j\right),$$
$$\tilde{\delta}_f = \tilde{\delta}_3\left((q_i, \mu_f(q_i)), a_k, q_j\right).$$

- *Membership Assignment Function:* The vague membership assignment function F_1^V is defined by the truth and falsity components of the neutrosophic membership assignment function F_1^N:

$$F_1^V = \left(F_1^\wedge, F_1^\vee\right),$$

where:

$$F_1^\wedge(\mu_t, \delta_t) = F_1^\wedge(\mu_1, \delta_1),$$
$$F_1^\vee(\mu_f, \delta_f) = F_1^\vee(\mu_3, \delta_3).$$

- *Multi-Membership Resolution Function:* The vague multi-membership resolution function F_2^V is derived from the neutrosophic resolution function F_2^N, retaining only the truth and falsity components:

$$F_2^V = \left(F_2^\wedge, F_2^\vee\right),$$

where:

$$F_2^\wedge\left(\{v_t^i\}\right) = F_2^\wedge\left(\{v_1^i\}\right),$$
$$F_2^\vee\left(\{v_f^i\}\right) = F_2^\vee\left(\{v_3^i\}\right).$$

By appropriately redefining the start states, transition functions, membership assignment, and resolution functions, a Neutrosophic General Finite Automaton (GNA) can be transformed into a Vague General Finite Automaton (VGFA). □

Theorem 2.22. *A Vague General Finite Automaton (VGFA) is a generalization of a General Fuzzy Automaton (GFA).*

Proof. Let $\tilde{F}_{\text{GFA}} = (Q, \Sigma, \tilde{R}, Z, \tilde{\delta}, \omega, F_1, F_2)$ be a General Fuzzy Automaton. Define a VGFA $\tilde{F}_{\text{VGFA}} = (Q, \Sigma, \tilde{R}_{\text{V}}, Z, \tilde{\delta}_{\text{V}}, \omega, F_1^{\text{V}}, F_2^{\text{V}})$ as follows:

- Use the same set of states, input symbols, and output symbols:

$$Q_{\text{VGFA}} = Q_{\text{GFA}}, \quad \Sigma_{\text{VGFA}} = \Sigma_{\text{GFA}}, \quad Z_{\text{VGFA}} = Z_{\text{GFA}}.$$

- Define the vague start states:

$$\tilde{R}_{\text{V}} = \left\{(q, \mu, 1-\mu) \mid \mu \in \tilde{R}\right\}.$$

- Extend the fuzzy transition function $\tilde{\delta}$ to the vague transition function $\tilde{\delta}_{\mathrm{V}}$:

$$\tilde{\delta}_{\mathrm{V}}((q,\mu_t,\mu_f),a,q') = \begin{cases} (\tilde{\delta}(q,a,q'), 1-\tilde{\delta}(q,a,q')) & \text{if } \tilde{\delta}(q,a,q') \neq 0, \\ (0,1) & \text{otherwise.} \end{cases}$$

- Define F_1^{V} to operate on truth and falsity components:

$$F_1^{\mathrm{V}}((\mu_t,\mu_f),(\delta_t,\delta_f)) = (\min(\mu_t,\delta_t), \max(\mu_f,\delta_f)).$$

- Use F_2^{V} to resolve multi-membership values:

$$F_2^{\mathrm{V}}(\{(\mu_{t,i},\mu_{f,i})\}) = \left(\max_i \mu_{t,i}, \min_i \mu_{f,i}\right).$$

By construction, $\tilde{F}_{\mathrm{VGFA}}$ reproduces the behavior of $\tilde{F}_{\mathrm{GFA}}$ when the falsity membership is the complement of truth membership. Thus, VGFA generalizes GFA. □

Theorem 2.23. *A Vague General Finite Automaton (VGFA) is a generalization of a Finite Automaton (FA).*

Proof. Let $A = (Q,\Sigma,\delta,q_0,F)$ be a finite automaton. Define a VGFA $\tilde{F}_{\mathrm{VGFA}} = (Q,\Sigma,\tilde{R}_{\mathrm{V}},Z,\tilde{\delta}_{\mathrm{V}},\omega,F_1^{\mathrm{V}},F_2^{\mathrm{V}})$ as follows:

- Use the same set of states and input symbols:

$$Q_{\mathrm{VGFA}} = Q_{\mathrm{FA}}, \quad \Sigma_{\mathrm{VGFA}} = \Sigma_{\mathrm{FA}}.$$

- Define the vague start states:

$$\tilde{R}_{\mathrm{V}} = \{(q,1,0) \text{ if } q = q_0,\ (q,0,1) \text{ otherwise.}\}.$$

- Define the vague transition function:

$$\tilde{\delta}_{\mathrm{V}}((q,\mu_t,\mu_f),a,q') = \begin{cases} (1,0) & \text{if } \delta(q,a) = q', \\ (0,1) & \text{otherwise.} \end{cases}$$

- Use trivial membership assignment functions:

$$F_1^{\mathrm{V}}((\mu_t,\mu_f),(\delta_t,\delta_f)) = (\delta_t,\delta_f).$$

- Define F_2^{V} as follows:

$$F_2^{\mathrm{V}}(\{(\mu_{t,i},\mu_{f,i})\}) = \left(\max_i \mu_{t,i}, \min_i \mu_{f,i}\right).$$

Under this construction:

(1) Each state in A maps to a vague state in $\tilde{F}_{\mathrm{VGFA}}$, with truth membership 1 and falsity membership 0 when active.

(2) The deterministic transitions of A are captured by the vague transition function with truth membership 1.

Thus, $\tilde{F}_{\mathrm{VGFA}}$ reproduces the behavior of A as a special case where truth and falsity membership values are binary. Therefore, VGFA generalizes FA. □

2.5. *Plithogenic General Finite Automaton*

Plithogenic General Finite Automaton represent a Plithogenic adaptation of General Finite Automata. The definitions and theorems, including related concepts, are provided below.

Definition 2.24. [156, 157] Let S be a universal set, and $P \subseteq S$. A *Plithogenic Set PS* is defined as:

$$PS = (P, v, Pv, pdf, pCF)$$

where:

- v is an attribute.
- Pv is the range of possible values for the attribute v.
- $pdf : P \times Pv \to [0,1]^s$ is the *Degree of Appurtenance Function (DAF).*
- $pCF : Pv \times Pv \to [0,1]^t$ is the *Degree of Contradiction Function (DCF).*

These functions satisfy the following axioms for all $a, b \in Pv$:

(1) *Reflexivity of Contradiction Function*:

$$pCF(a, a) = 0$$

(2) *Symmetry of Contradiction Function*:

$$pCF(a, b) = pCF(b, a)$$

Example 2.25. (cf. [61]) Here, $s, t \in \{1, 2, 3, 4\}$.

- When $s = t = 1$, PS is called a *Plithogenic Fuzzy Set* and is denoted by PFS.
- When $s = 2, t = 1$, PS is called a *Plithogenic Intuitionistic Fuzzy Set* and is denoted by $PIFS$.
- When $s = 3, t = 1$, PS is called a *Plithogenic Neutrosophic Set* and is denoted by PNS.
- When $s = 4, t = 1$, PS is called a *Plithogenic Turiyam Neutrosophic Set* and is denoted by $PTuS$.

Definition 2.26. A *Plithogenic General Finite Automaton (PGFA)* is defined as an eight-tuple:

$$\tilde{F} = (Q, \Sigma, \tilde{R}, Z, \tilde{\delta}, \omega, F_1, F_2),$$

where:

(1) *States*:

- Q is a finite set of states:

$$Q = \{q_1, q_2, \ldots, q_n\}.$$

(2) *Input Symbols*:

- Σ is a finite set of input symbols:

$$\Sigma = \{a_1, a_2, \ldots, a_m\}.$$

(3) *Plithogenic Start States*:

- $\tilde{R}$ is the set of plithogenic start states:

$$\tilde{R} = \{(q, v_q, \mathrm{DAF}_R(q, v_q)) \mid q \in R\},$$

where:

  - $R \subseteq Q$ is the set of initial states.
  - $v_q \in V$ is the attribute value of state q.
  - $\mathrm{DAF}_R(q, v_q) \in [0,1]^s$ is the Degree of Appurtenance Function (DAF) for state q.

(4) *Output Symbols*:

- Z is a finite set of output symbols:

$$Z = \{b_1, b_2, \ldots, b_k\}.$$

(5) *Plithogenic Transition Function*:

- $\tilde{\delta} : (Q \times V \times [0,1]^s) \times \Sigma \times Q \to [0,1]^s$ is the augmented plithogenic transition function, defined by:

$$\tilde{\delta}\left((q_i, v_{q_i}, \mu(q_i)), a_k, q_j\right) = \mathrm{DAF}_\delta\left((q_i, q_j), (v_{q_i}, v_{q_j}), a_k\right),$$

where:

  - $\mu(q_i)$ is the Degree of Appurtenance of state q_i.
  - $v_{q_i}, v_{q_j} \in V$ are the attribute values of states q_i and q_j.
  - DAF_δ is determined using the Degree of Contradiction Function (DCF) and the DAF of the states.

(6) *Output Function*:

- $\omega : Q \to Z$ is the output function.

(7) *Membership Assignment Function*:

- $F_1 : [0,1]^s \times [0,1]^s \to [0,1]^s$ assigns the degree of appurtenance for transitions, considering the DAF and DCF.

(8) *Multi-Membership Resolution Function*:

- $F_2 : ([0,1]^s)^n \to [0,1]^s$ resolves multiple degrees of appurtenance when there are simultaneous transitions to the same state.

The degree of appurtenance of state q_j at time $t+1$ is calculated as:

$$\mu_{t+1}(q_j) = F_2\left(\left\{F_1\left(\mu_t(q_i), \mathrm{DAF}_\delta\left((q_i, q_j), (v_{q_i}, v_{q_j}), a_k\right)\right)\right\}_{i=1}^n\right),$$

where:

- n is the number of transitions leading to q_j at time $t+1$.
- $\mu_t(q_i)$ is the degree of appurtenance of state q_i at time t.
- $\mathrm{DAF}_\delta\left((q_i, q_j), (v_{q_i}, v_{q_j}), a_k\right)$ incorporates the attribute values and degrees of contradiction.

Theorem 2.27. *A Plithogenic General Finite Automaton (PGFA) can be transformed into a Neutrosophic General Finite Automaton (GNA), a Vague General Finite Automaton (VGFA), a Fuzzy General Finite Automaton (GFA), and a Turiyam Neutrosophic General Finite Automaton (TGFA) by appropriately selecting the parameters s and t in the Plithogenic set and redefining its transition and membership functions accordingly.*

Proof. A Plithogenic General Finite Automaton is defined using Plithogenic sets, which generalize various types of fuzzy sets by introducing an attribute with possible values and corresponding degrees of appurtenance and contradiction. By specializing the parameters s and t in the Plithogenic set, we can obtain different types of automata.

A *Plithogenic Set PS* is defined as:

$$PS = (P, v, Pv, \mathrm{DAF}, \mathrm{DCF}),$$

.

By choosing specific values for s and t, the Plithogenic set reduces to various fuzzy sets:

- $s = t = 1$: Plithogenic Fuzzy Set (PFS).
- $s = 2, t = 1$: Plithogenic Intuitionistic Fuzzy Set (PIFS).
- $s = 3, t = 1$: Plithogenic Neutrosophic Set (PNS).
- $s = 4, t = 1$: Plithogenic Turiyam Neutrosophic Set (PTuS).
- $s = 2, t = 1$ with additional constraints: Vague Set.

A *Plithogenic General Finite Automaton (PGFA)* is defined as an eight-tuple:

$$\tilde{F} = (Q, \Sigma, \tilde{R}, Z, \tilde{\delta}, \omega, F_1, F_2),$$

with states, input symbols, plithogenic start states, output symbols, plithogenic transition function, output function, membership assignment function, and multi-membership resolution function.

We will show that by selecting appropriate values of s and t and redefining functions, a PGFA can be transformed into each of the other automata.

We consider Transformation to a Fuzzy General Finite Automaton (GFA). Set $s = t = 1$. The Plithogenic set becomes a Plithogenic Fuzzy Set (PFS). The Degree of Appurtenance Function (DAF) and Degree of Contradiction Function (DCF) reduce to functions with a single value.

- *States*: Remain the same.
- *Start States*: The DAF now assigns a single membership degree $\mu(q) \in [0, 1]$ to each state q.
- *Transition Function*: The DAF for transitions assigns a single membership degree to each transition, similar to the transition weights in a GFA.
- *Membership Assignment Function* F_1: Becomes $F_1 : [0, 1] \times [0, 1] \to [0, 1]$.
- *Multi-Membership Resolution Function* F_2: Becomes $F_2 : [0, 1]^n \to [0, 1]$.

Thus, the PGFA reduces to a GFA, with the fuzzy membership degrees being the degrees of appurtenance from the Plithogenic set.

We consider Transformation to a Neutrosophic General Finite Automaton (GNA). Set $s = 3$, $t = 1$. The Plithogenic set becomes a Plithogenic Neutrosophic Set (PNS).

- *States*: Remain the same.
- *Start States*: The DAF assigns a vector $(\mu_T(q), \mu_I(q), \mu_F(q)) \in [0, 1]^3$ to each state q, representing truth, indeterminacy, and falsity membership degrees.
- *Transition Function*: The DAF for transitions assigns a vector of degrees to each transition.
- *Membership Assignment Function* F_1: Becomes $F_1 : [0, 1]^3 \times [0, 1]^3 \to [0, 1]^3$, handling the neutrosophic components.
- *Multi-Membership Resolution Function* F_2: Becomes $F_2 : ([0, 1]^3)^n \to [0, 1]^3$.

Thus, the PGFA reduces to a GNA, with the neutrosophic membership degrees derived from the degrees of appurtenance in the Plithogenic set.

We consider Transformation to a Vague General Finite Automaton (VGFA). Set $s = 2$, $t = 1$, and ensure that for each element x:

$$0 \leq \mu_T(x) + \mu_F(x) \leq 1.$$

This corresponds to a Vague Set.

- *States*: Remain the same.
- *Start States*: The DAF assigns a pair $(\mu_T(q), \mu_F(q)) \in [0, 1]^2$ to each state q, representing truth and falsity membership degrees.

- *Transition Function*: The DAF for transitions assigns pairs of degrees to each transition.
- *Membership Assignment Function* F_1: Becomes $F_1 : [0,1]^2 \times [0,1]^2 \to [0,1]^2$.
- *Multi-Membership Resolution Function* F_2: Becomes $F_2 : ([0,1]^2)^n \to [0,1]^2$.

Thus, the PGFA reduces to a VGFA, with the vague membership degrees obtained from the degrees of appurtenance in the Plithogenic set.

We consider Transformation to a Turiyam Neutrosophic General Finite Automaton (TGFA). Set $s = 4$, $t = 1$. The Plithogenic set becomes a Plithogenic Turiyam Neutrosophic Set (PTuS).

- *States*: Remain the same.
- *Start States*: The DAF assigns a vector $(\mu_T(q), \mu_I(q), \mu_F(q), \mu_L(q)) \in [0,1]^4$ to each state q, representing truth, indeterminacy, falsity, and liberal membership degrees.
- *Transition Function*: The DAF for transitions assigns a vector of degrees to each transition.
- *Membership Assignment Function* F_1: Becomes $F_1 : [0,1]^4 \times [0,1]^4 \to [0,1]^4$.
- *Multi-Membership Resolution Function* F_2: Becomes $F_2 : ([0,1]^4)^n \to [0,1]^4$.

Thus, the PGFA reduces to a TGFA, with the Turiyam Neutrosophic membership degrees derived from the degrees of appurtenance in the Plithogenic set.

By selecting appropriate values for s and t in the Plithogenic set and redefining the functions accordingly, a Plithogenic General Finite Automaton can be transformed into a Neutrosophic, Vague, Fuzzy, or Turiyam Neutrosophic General Finite Automaton. □

3. Graph Grammar

Graph Grammar is a formal framework that defines rules for graph transformations, enabling the representation of dynamic changes in graph structures.We explore the concepts of Graph Grammar within the contexts of Fuzzy, Neutrosophic, Vague, Turiyam, and Plithogenic frameworks. First, we present the fundamental definitions of Graph Grammar. The definitions are provided below.

Definition 3.1 (Graph)**.** [41] A *graph* G is a mathematical structure that represents relationships between objects. It consists of a set of vertices $V(G)$ and a set of edges $E(G)$, where each edge connects a pair of vertices. Formally, a graph is represented as $G = (V, E)$, where V is the set of vertices and E is the set of edges.

Definition 3.2 (Degree)**.** [41] Let $G = (V, E)$ be a graph. The *degree* of a vertex $v \in V$, denoted $\deg(v)$, is defined as the number of edges connected to v. For undirected graphs, the degree is given by:

$$\deg(v) = |\{e \in E \mid v \in e\}|.$$

For directed graphs, the *in-degree* $\deg^-(v)$ refers to the number of edges directed towards v, while the *out-degree* $\deg^+(v)$ represents the number of edges directed away from v.

Definition 3.3. (cf. [137]) An *empty graph* $G = (V, E)$ is a graph with a non-empty set of vertices V and an empty set of edges E. Formally:

$$G = (V, \emptyset),$$

where $V \neq \emptyset$ and $E = \emptyset$.

Definition 3.4. A *null graph* $G = (V, E)$ is a graph with an empty vertex set V and consequently an empty edge set E. Formally:

$$G = (\emptyset, \emptyset).$$

Definition 3.5 (Graph Grammar)**.** (cf. [48, 48, 128]) Graph Grammar is a formal framework for representing dynamic transformations of graph structures. It defines a set of rules for graph rewriting and consists of the following components:

(1) *Vertex and Edge Labels:*
- Vertex label set: $\Sigma_V = N_V \cup T_V$, where:
  - N_V: Non-terminal labels.
  - T_V: Terminal labels.
- Edge label set: $\Sigma_E = N_E \cup T_E$, where:
  - N_E: Non-terminal labels.
  - T_E: Terminal labels.

(2) *Start Graph:* The start graph G_S serves as the initial graph for transformations.

(3) *Production Rules:* A finite set of production rules P defines the transformations, where each rule $p \in P$ is represented as:

$$p : G_L \xleftarrow{l} H \xrightarrow{r} G_R,$$

where:
- G_L: Left-hand side graph, representing the pattern to match.
- G_R: Right-hand side graph, representing the replacement graph.
- H: Interface graph, representing the common subgraph used for gluing G_L and G_R.

(4) *Isomorphic Mappings:* The interface graph H is associated with the following label-preserving isomorphisms:

$$\phi_L : H' \to \phi_L(H) \subseteq G_L, \quad \phi_R : H' \to \phi_R(H) \subseteq G_R,$$

where:
- ϕ_L: Maps H to G_L.

- ϕ_R: Maps H to G_R.

Remark 3.6. A production rule p searches for a subgraph in G isomorphic to G_L. Upon matching, it replaces G_L with G_R, preserving the common structure H by gluing G_R at the interface defined by H.

Example 3.7 (Simple Graph Grammar)**.** Consider a simple Graph Grammar GG $= (T, G_0, \text{Rule})$:

- *Type Graph* T:
  - Vertex labels: $\Sigma_V = \{A, B\}$.
  - Edge labels: $\Sigma_E = \{a\}$.
- *Start Graph* G_0:
  - Vertices: $V = \{v_1\}$ with label A.
  - Edges: None.
- *Production Rule* p:

$$p : G_L \xleftarrow{l} H \xrightarrow{r} G_R,$$

  where:
  - G_L:
    - Vertices: $\{v\}$ with label A.
    - Edges: None.
  - G_R:
    - Vertices: $\{v_1, v_2\}$ with labels A, B, respectively.
    - Edges: $\{e\}$ connecting v_1 to v_2 with label a.
  - H:
    - Vertices: $\{v\}$ with label A.
    - Edges: None.
  - Mappings:
    - $l : H \to G_L$, identity on v.
    - $r : H \to G_R$, maps v to v_1.

Applying the production rule p to G_0 results in a graph with one A-labeled vertex connected by an a-labeled edge to a new B-labeled vertex.

Theorem 3.8. *Graph Grammar can represent both the empty graph and the null graph.*

Proof. Graph Grammar provides a framework for defining graph structures and their transformations through a set of rules. We show that both the empty graph and the null graph can be represented using Graph Grammar.

The empty graph $G = (V, E)$ is defined as having a non-empty vertex set V and an empty edge set $E = \emptyset$. Using Graph Grammar, we can construct G as follows:

(1) **Type Graph T:** Define a vertex label set $\Sigma_V = \{A\}$ and an edge label set $\Sigma_E = \emptyset$.
(2) **Start Graph G_S:** Initialize G_S with vertices $V = \{v_1, v_2, \ldots, v_n\}$, where $n > 0$, and $E = \emptyset$.
(3) **Production Rules:** No rules are needed, as the empty graph does not include any edges or transformations.

Thus, G_S itself represents the empty graph.

The null graph $G = (\emptyset, \emptyset)$ is defined as having no vertices and no edges. Using Graph Grammar, we can construct G as follows:

(1) **Type Graph T:** Define an empty vertex label set $\Sigma_V = \emptyset$ and an edge label set $\Sigma_E = \emptyset$.
(2) **Start Graph G_S:** Initialize $G_S = (\emptyset, \emptyset)$.
(3) **Production Rules:** No rules are needed, as the null graph is static and requires no transformations.

Thus, $G_S = (\emptyset, \emptyset)$ directly represents the null graph.

Graph Grammar can represent both the empty graph and the null graph by appropriately defining the vertex and edge label sets, the start graph, and the (optional) production rules. □

3.1. *Fuzzy Graph Grammar*

Fuzzy Graph Grammar is an extension of Graph Grammar with added fuzzy conditions, and it has been explored in several studies [22, 79, 118, 120, 183]. The definitions, including related concepts, are provided below.

Definition 3.9. [127] A *fuzzy graph* G is defined as:

$$G = ((V_G, \mathrm{pert}_V), (E_G, \mathrm{pert}_E), \mathrm{source}_G, \mathrm{target}_G),$$

where:

- V_G is a fuzzy set of vertices with a membership function $\mathrm{pert}_V : V_G \to [0, 1]$, which assigns a membership degree to each vertex.
- E_G is a fuzzy set of edges with a membership function $\mathrm{pert}_E : E_G \to [0, 1]$, which assigns a membership degree to each edge.
- $\mathrm{source}_G : E_G \to V_G$ is the *source mapping* of edges.
- $\mathrm{target}_G : E_G \to V_G$ is the *target mapping* of edges.

The membership degrees $\mathrm{pert}_V(v)$ and $\mathrm{pert}_E(e)$ indicate the presence or influence of vertices $v \in V_G$ and edges $e \in E_G$, respectively.

Definition 3.10. [22] Let

$$G = ((V_G, \mathrm{pert}_V), (E_G, \mathrm{pert}_E), \mathrm{source}_G, \mathrm{target}_G)$$

and

$$H = \big((V_H, \mathrm{pert}'_V), (E_H, \mathrm{pert}'_E), \mathrm{source}_H, \mathrm{target}_H\big)$$

be two fuzzy graphs. A *fuzzy graph morphism* $f : G \to H$ is a pair of fuzzy set morphisms (f_V, f_E) such that:

$$f_V \circ \mathrm{source}_G = \mathrm{source}_H \circ f_E \quad \text{and} \quad f_V \circ \mathrm{target}_G = \mathrm{target}_H \circ f_E,$$

where:

- $f_V : V_G \to V_H$ maps vertices from G to H.
- $f_E : E_G \to E_H$ maps edges from G to H.
- The mappings f_V and f_E must preserve the source and target of edges, as well as the membership degrees.

Definition 3.11. [22] A *typed fuzzy graph* G_T is defined as a tuple:

$$G_T = (G, t_G, T),$$

where:

- G is a fuzzy graph.
- T is a *fuzzy type graph* that characterizes the types of vertices and edges allowed in the grammar.
- $t_G : G \to T$ is a total fuzzy graph morphism, ensuring that each fuzzy graph in the grammar conforms to the fuzzy type graph T.

Definition 3.12. [22] A *fuzzy rule* p in FGG is defined as a tuple:

$$p : L \xleftarrow{l} K \xrightarrow{r} R,$$

where:

- L, K, R are fuzzy graphs typed over the fuzzy type graph T.
- l and r are total fuzzy graph morphisms, representing the left-hand side, interface, and right-hand side of the rule, respectively.

The following sets are associated with the application of a fuzzy rule:

- *Deleted vertices:* $\mathrm{RuleDel}_V^p = V_L \setminus \mathrm{rng}(l_V)$.
- *Deleted edges:* $\mathrm{RuleDel}_E^p = E_L \setminus \mathrm{rng}(l_E)$.
- *Preserved vertices:* $\mathrm{RulePres}_V^p = V_L \setminus \mathrm{RuleDel}_V^p$.
- *Preserved edges:* $\mathrm{RulePres}_E^p = E_L \setminus \mathrm{RuleDel}_E^p$.
- *Created vertices:* $\mathrm{RuleCreated}_V^p = V_R \setminus \mathrm{rng}(r_V)$.

- *Created edges:* $\text{RuleCreated}_E^p = E_R \setminus \text{rng}(r_E)$.

Definition 3.13. [22] A *Typed Fuzzy Graph Grammar (FGG)* is defined as a tuple:

$$\text{FGG} = (T, G_0, \text{Rule}),$$

where:

- T is the fuzzy type graph.
- G_0 is the initial fuzzy graph typed over T, representing the initial state of the system.
- Rule is a set of fuzzy rules that describe the state transitions allowed within the system.

The behavior of a fuzzy graph grammar is determined by the application of these fuzzy rules, enabling the transition from one fuzzy graph to another by modifying vertices and edges according to the specified rules.

Example 3.14 (Fuzzy Graph Grammar)**.** Consider a Fuzzy Graph Grammar $\text{FGG} = (T', G_0', \text{Rule}')$:

- *Type Graph T'*:
  - Vertices with membership functions pert_V:
    - Vertex A with $\text{pert}_V(A) = 1$.
    - Vertex B with $\text{pert}_V(B) = 1$.
  - Edges with membership functions pert_E:
    - Edge a with $\text{pert}_E(a) = 1$.
- *Start Graph G_0'*:
  - Vertices: $V = \{v_1\}$ with label A and $\text{pert}_V(v_1) = 0.8$.
  - Edges: None.
- *Fuzzy Production Rule p'*:

$$p' : L \xleftarrow{l} K \xrightarrow{r} R,$$

  where:
  - L:
    - Vertices: $\{v\}$ with label A and $\text{pert}_V(v) = \alpha$.
    - Edges: None.
  - R:
    - Vertices: $\{v_1, v_2\}$ with labels A, B and $\text{pert}_V(v_1) = \alpha$, $\text{pert}_V(v_2) = \beta$.
    - Edges: $\{e\}$ connecting v_1 to v_2 with label a and $\text{pert}_E(e) = \gamma$.
  - K:
    - Vertices: $\{v\}$ with label A and $\text{pert}_V(v) = \alpha$.
    - Edges: None.
  - Mappings:

* $l : K \to L$, identity on v.
* $r : K \to R$, maps v to v_1.

Applying the fuzzy production rule p' to G'_0 results in:

- The original vertex v_1 retains its membership degree $\text{pert}_V(v_1) = 0.8$.
- A new vertex v_2 with label B and membership degree $\text{pert}_V(v_2) = 0.6$ (assuming $\beta = 0.6$).
- A new edge e connecting v_1 and v_2 with membership degree $\text{pert}_E(e) = 0.7$ (assuming $\gamma = 0.7$).

The membership degrees represent the uncertainty or partial presence of vertices and edges in the fuzzy graph. This example illustrates how Fuzzy Graph Grammar extends Graph Grammar by incorporating fuzzy membership values.

Theorem 3.15. *A Fuzzy Graph Grammar generalizes a Graph Grammar by allowing vertices and edges to have membership degrees between 0 and 1. A standard Graph Grammar is a special case of a Fuzzy Graph Grammar where all membership degrees are either 0 or 1.*

Proof. Let $\text{GG} = (T, G_0, \text{Rule})$ be a Graph Grammar, where:

- Vertices and edges are unweighted (membership degrees of 1 for existing elements, and implicitly 0 for non-existing elements).
- T is the type graph, G_0 is the initial graph, and Rule is the set of production rules.

We can construct a corresponding Fuzzy Graph Grammar $\text{FGG} = (T', G'_0, \text{Rule}')$ as follows:

- For each vertex v in T, set $\text{pert}_V(v) = 1$ in T'.
- For each edge e in T, set $\text{pert}_E(e) = 1$ in T'.
- Similarly, define G'_0 and Rule' by assigning membership degrees of 1 to all vertices and edges present in G_0 and Rule.

In this construction, the fuzzy membership functions pert_V and pert_E take only values in $\{0, 1\}$. Therefore, the behavior of FGG replicates that of GG, demonstrating that Graph Grammar is a special case of Fuzzy Graph Grammar. Hence, Fuzzy Graph Grammar generalizes Graph Grammar by allowing membership degrees in the interval $[0, 1]$. □

Theorem 3.16. *A Fuzzy Graph Grammar can represent both the empty graph and the null graph.*

Proof. Fuzzy Graph Grammar extends Graph Grammar by allowing vertices and edges to have membership degrees in the interval $[0, 1]$. We demonstrate how a Fuzzy Graph Grammar can represent both the empty graph and the null graph.

The empty graph $G = (V, E)$ has a non-empty vertex set V and an empty edge set $E = \emptyset$.

Using Fuzzy Graph Grammar, we define:

(1) **Type Graph T:** Define a vertex label set $\Sigma_V = \{A\}$ and an edge label set $\Sigma_E = \emptyset$.
(2) **Initial Fuzzy Graph G_0:** Initialize G_0 with:
   - A fuzzy vertex set $V_G = \{v_1, v_2, \ldots, v_n\}$ with membership function $\mathrm{pert}_V(v_i) = 1$ for all v_i.
   - A fuzzy edge set $E_G = \emptyset$.
(3) **Production Rules:** No rules are needed, as the empty graph contains no edges or transformations.

Since the membership function assigns $\mathrm{pert}_E(e) = 0$ for all edges e, the resulting fuzzy graph G_0 represents the empty graph.

The null graph $G = (\emptyset, \emptyset)$ has both an empty vertex set and an empty edge set.

Using Fuzzy Graph Grammar, we define:

(1) **Type Graph T:** Define an empty vertex label set $\Sigma_V = \emptyset$ and an edge label set $\Sigma_E = \emptyset$.
(2) **Initial Fuzzy Graph G_0:** Initialize G_0 with:
   - $V_G = \emptyset$.
   - $E_G = \emptyset$.
(3) **Production Rules:** No rules are needed, as the null graph is static.

Since $V_G = \emptyset$ and $E_G = \emptyset$, the resulting fuzzy graph G_0 directly represents the null graph.

Fuzzy Graph Grammar can represent both the empty graph and the null graph by appropriately defining the type graph, the initial fuzzy graph, and the membership functions. This demonstrates the expressiveness of Fuzzy Graph Grammar in capturing these fundamental graph structures. □

3.2. *Neutrosophic Graph Grammar*

Neutrosophic Graph Grammar is an extension of Graph Grammar with added Neutrosophic conditions. The definitions and theorems, including related concepts, are provided below.

Definition 3.17. [169] A *Neutrosophic Graph* G is defined as a quadruple:

$$G = (V, E, \sigma, \mu),$$

where:

- V is a finite set of vertices.
- E is a set of edges, where each edge connects two vertices in V.

- $\sigma : V \to [0,1]^3$ assigns to each vertex $v \in V$ a triple $\sigma(v) = (\sigma_T(v), \sigma_I(v), \sigma_F(v))$, representing the *truth-membership*, *indeterminacy-membership*, and *falsity-membership* degrees, respectively.
- $\mu : E \to [0,1]^3$ assigns to each edge $e \in E$ a triple $\mu(e) = (\mu_T(e), \mu_I(e), \mu_F(e))$.

These membership degrees satisfy the condition for all $v \in V$ and $e \in E$:

$$0 \leq \sigma_T(v) + \sigma_I(v) + \sigma_F(v) \leq 3, \quad 0 \leq \mu_T(e) + \mu_I(e) + \mu_F(e) \leq 3.$$

Additionally, for every edge $e = (v_i, v_j) \in E$:

$$\mu_T(e) \leq \min\{\sigma_T(v_i), \sigma_T(v_j)\}.$$

Definition 3.18. Let $G = (V_G, E_G, \sigma_G, \mu_G)$ and $H = (V_H, E_H, \sigma_H, \mu_H)$ be two neutrosophic graphs. A *Neutrosophic Graph Morphism* $f : G \to H$ is a pair of functions (f_V, f_E) where:

- $f_V : V_G \to V_H$ maps vertices of G to vertices of H.
- $f_E : E_G \to E_H$ maps edges of G to edges of H.
- The mappings preserve the incidence relations:

$$f_V(\text{source}_G(e)) = \text{source}_H(f_E(e)), \quad f_V(\text{target}_G(e)) = \text{target}_H(f_E(e)), \quad \forall e \in E_G.$$

- The membership degrees satisfy:

$$\sigma_G(v) \leq \sigma_H(f_V(v)), \quad \mu_G(e) \leq \mu_H(f_E(e)), \quad \forall v \in V_G, e \in E_G.$$

Definition 3.19. A *Typed Neutrosophic Graph* G_T is a triple:

$$G_T = (G, t_G, T),$$

where:

- G is a neutrosophic graph.
- T is a *Neutrosophic Type Graph* specifying allowed types and membership degrees for vertices and edges.
- $t_G : G \to T$ is a total neutrosophic graph morphism, ensuring that G conforms to the type graph T.

Definition 3.20. A *Neutrosophic Rule* p is defined as:

$$p : L \xleftarrow{l} K \xrightarrow{r} R,$$

where:

- L, K, R are neutrosophic graphs typed over the neutrosophic type graph T.
- l and r are total neutrosophic graph morphisms.

The following sets are associated with the rule p:

- *Deleted Vertices*: $\text{Del}_V^p = V_L \setminus \text{Im}(l_V)$.

- *Deleted Edges*: $\mathrm{Del}_E^p = E_L \setminus \mathrm{Im}(l_E)$.
- *Preserved Vertices*: $\mathrm{Pres}_V^p = \mathrm{Im}(l_V)$.
- *Preserved Edges*: $\mathrm{Pres}_E^p = \mathrm{Im}(l_E)$.
- *Created Vertices*: $\mathrm{Cre}_V^p = V_R \setminus \mathrm{Im}(r_V)$.
- *Created Edges*: $\mathrm{Cre}_E^p = E_R \setminus \mathrm{Im}(r_E)$.

Definition 3.21. A *Typed Neutrosophic Graph Grammar (NGG)* is a triple:

$$\mathrm{NGG} = (T, G_0, \mathrm{Rule}),$$

where:

- T is the neutrosophic type graph.
- G_0 is the initial neutrosophic graph typed over T.
- Rule is a set of neutrosophic rules.

The grammar defines the possible transformations of neutrosophic graphs through the application of rules, modeling the dynamic behavior of systems with neutrosophic uncertainty.

Theorem 3.22. *A Neutrosophic Graph Grammar generalizes a Fuzzy Graph Grammar by allowing vertices and edges to have truth, indeterminacy, and falsity membership values, whereas a Fuzzy Graph Grammar is limited to truth-membership only.*

Proof. Let $\mathrm{FGG} = (T, G_0, \mathrm{Rule})$ be a Fuzzy Graph Grammar where:

- Vertices and edges have fuzzy membership values $\mathrm{pert}_V : V \to [0,1]$ and $\mathrm{pert}_E : E \to [0,1]$.

Define a corresponding Neutrosophic Graph Grammar $\mathrm{NGG} = (T, G_0, \mathrm{Rule})$ where:

- Each vertex $v \in V$ and edge $e \in E$ is assigned neutrosophic membership values $\sigma(v) = (\sigma_T(v), \sigma_I(v), \sigma_F(v))$ and $\mu(e) = (\mu_T(e), \mu_I(e), \mu_F(e))$, satisfying:

$$0 \leq \sigma_T(v) + \sigma_I(v) + \sigma_F(v) \leq 3, \quad 0 \leq \mu_T(e) + \mu_I(e) + \mu_F(e) \leq 3.$$

When $\sigma_I(v) = 0$ and $\sigma_F(v) = 0$ for all $v \in V$, and similarly $\mu_I(e) = 0$ and $\mu_F(e) = 0$ for all $e \in E$, the NGG reduces to the FGG. Therefore, NGG generalizes FGG by introducing indeterminacy and falsity memberships, enabling richer representation of uncertainty. □

Corollary 3.23. *A Neutrosophic Graph Grammar can represent both the empty graph and the null graph.*

Proof. The proof follows almost the same method as for the Fuzzy Graph Grammar. □

3.3. *Turiyam Neutrosophic Graph Grammar*

Turiyam Neutrosophic Graph Grammar is an extension of Graph Grammar with added Turiyam Neutrosophic conditions. The definitions and theorems, including related concepts, are provided below.

Definition 3.24 (Turiyam Neutrosophic Graph)**.** [69–71,144] A *Turiyam Neutrosophic Graph* G^T is defined as:

$$G^T = (V^T, E^T),$$

where:

- V^T is the set of vertices with Turiyam Neutrosophic membership functions.
- E^T is the set of edges with Turiyam Neutrosophic membership functions.

For each vertex $v \in V^T$, there are mappings:

$$T(v), I(v), F(v), L(v) : V^T \to [0, 1],$$

where:

- $T(v)$ is the truth-membership degree.
- $I(v)$ is the indeterminacy-membership degree.
- $F(v)$ is the falsity-membership degree.
- $L(v)$ is the liberal state-membership degree.

These degrees satisfy:

$$0 \leq T(v) + I(v) + F(v) + L(v) \leq 4, \quad \forall v \in V^T.$$

Similarly, for each edge $e = (v_i, v_j) \in E^T$, there are mappings:

$$T(e), I(e), F(e), L(e) : E^T \to [0, 1],$$

satisfying:

$$0 \leq T(e) + I(e) + F(e) + L(e) \leq 4, \quad \forall e \in E^T.$$

Definition 3.25. Let $G = (V_G^T, E_G^T)$ and $H = (V_H^T, E_H^T)$ be two Turiyam Neutrosophic graphs. A *Turiyam Neutrosophic Graph Morphism* $f : G \to H$ is a pair of functions (f_V, f_E) where:

- $f_V : V_G^T \to V_H^T$.
- $f_E : E_G^T \to E_H^T$.
- The mappings preserve the incidence relations:

$$f_V(\text{source}_G(e)) = \text{source}_H(f_E(e)), \quad f_V(\text{target}_G(e)) = \text{target}_H(f_E(e)), \quad \forall e \in E_G^T.$$

- The Turiyam Neutrosophic membership degrees satisfy:

$$T_G(v) \leq T_H(f_V(v)), \quad I_G(v) \geq I_H(f_V(v)),$$
$$F_G(v) \geq F_H(f_V(v)), \quad L_G(v) \geq L_H(f_V(v)), \quad \forall v \in V_G^T,$$
$$T_G(e) \leq T_H(f_E(e)), \quad \text{and similarly for } I, F, L.$$

Definition 3.26. A *Typed Turiyam Neutrosophic Graph* G_T^T is a triple:

$$G_T^T = (G^T, t_G, T^T),$$

where:

- G^T is a Turiyam Neutrosophic graph.
- T^T is a *Turiyam Neutrosophic Type Graph* specifying allowed types and Turiyam Neutrosophic membership degrees.
- $t_G : G^T \to T^T$ is a total Turiyam Neutrosophic graph morphism.

Definition 3.27. A *Turiyam Neutrosophic Rule* p is defined as:

$$p : L^T \xleftarrow{l} K^T \xrightarrow{r} R^T,$$

where:

- L^T, K^T, R^T are Turiyam Neutrosophic graphs typed over the Turiyam Neutrosophic type graph T^T.
- l and r are total Turiyam Neutrosophic graph morphisms.

Definition 3.28. A *Typed Turiyam Neutrosophic Graph Grammar (TGG)* is a triple:

$$\text{TGG} = (T^T, G_0^T, \text{Rule}),$$

where:

- T^T is the Turiyam Neutrosophic type graph.
- G_0^T is the initial Turiyam Neutrosophic graph typed over T^T.
- Rule is a set of Turiyam Neutrosophic rules.

The grammar defines transformations of Turiyam Neutrosophic graphs, capturing systems with Turiyam Neutrosophic uncertainty involving truth, indeterminacy, falsity, and liberal states.

Theorem 3.29. *A Turiyam Neutrosophic Graph Grammar (TGG) can be transformed into a Neutrosophic Graph Grammar (NGG) and a Fuzzy Graph Grammar (FGG) by appropriately redefining the membership functions to match the specific requirements of each grammar type.*

Proof. Turiyam Neutrosophic Graph Grammar extends the conventional graph grammar by incorporating four membership components: truth, indeterminacy, falsity, and liberal state.

By selectively adjusting or omitting these components, we can transform a TGG into either an NGG or an FGG.

A *Turiyam Neutrosophic Graph* G^T is defined as:

$$G^T = (V^T, E^T),$$

where:

- V^T is the set of vertices with Turiyam Neutrosophic membership functions.
- E^T is the set of edges with Turiyam Neutrosophic membership functions.

For each vertex $v \in V^T$, there are mappings:

$$T(v), I(v), F(v), L(v) : V^T \to [0, 1].$$

Similarly, for each edge $e = (v_i, v_j) \in E^T$, there are mappings:

$$T(e), I(e), F(e), L(e) : E^T \to [0, 1].$$

These membership degrees satisfy:

$$0 \leq T(v) + I(v) + F(v) + L(v) \leq 4, \quad 0 \leq T(e) + I(e) + F(e) + L(e) \leq 4.$$

We will show how a TGG can be transformed into an NGG or an FGG by redefining the membership functions and adjusting the grammar rules accordingly.

We consider Transformation to a Neutrosophic Graph Grammar (NGG). To transform a TGG into an NGG, we set the liberal state membership degree to zero:

$$L(v) = 0, \quad L(e) = 0, \quad \forall v \in V^T,\, e \in E^T.$$

The Turiyam Neutrosophic graph G^T becomes a Neutrosophic Graph G, where each vertex v and edge e is represented by a triple of membership degrees:

$$\sigma(v) = (T(v), I(v), F(v)), \quad \mu(e) = (T(e), I(e), F(e)).$$

- The truth-membership, indeterminacy-membership, and falsity-membership components are preserved.
- The conditions for a Neutrosophic Graph are satisfied:

$$0 \leq \sigma_T(v) + \sigma_I(v) + \sigma_F(v) \leq 3, \quad 0 \leq \mu_T(e) + \mu_I(e) + \mu_F(e) \leq 3.$$

- The rules and morphisms of the Turiyam Neutrosophic graph grammar are adjusted by ignoring the liberal state component, resulting in the structure of an NGG.

We consider Transformation to a Fuzzy Graph Grammar (FGG). To transform a TGG into an FGG, we retain only the truth-membership component:

$$I(v) = 0, \quad F(v) = 0, \quad L(v) = 0, \quad \forall v \in V^T,$$
$$I(e) = 0, \quad F(e) = 0, \quad L(e) = 0, \quad \forall e \in E^T.$$

The Turiyam Neutrosophic graph G^T becomes a Fuzzy Graph G, where each vertex v and edge e is represented by a single membership degree:

$$\mathrm{pert}_V(v) = T(v), \quad \mathrm{pert}_E(e) = T(e).$$

- The truth-membership degree serves as the fuzzy membership degree for vertices and edges.
- The conditions for a Fuzzy Graph are satisfied:

$$0 \leq \mathrm{pert}_V(v) \leq 1, \quad 0 \leq \mathrm{pert}_E(e) \leq 1.$$

- The rules and morphisms of the Turiyam Neutrosophic graph grammar are adjusted to include only the truth-membership component, resulting in the structure of an FGG.

By redefining the membership components and adjusting the grammar rules, a Turiyam Neutrosophic Graph Grammar can be transformed into either a Neutrosophic Graph Grammar or a Fuzzy Graph Grammar, depending on which components are retained or omitted. □

Corollary 3.30. *A Turiyam Neutrosophic Graph Grammar can represent both the empty graph and the null graph.*

Proof. The proof follows almost the same method as for the Fuzzy Graph Grammar. □

3.4. *Vague Graph Grammar*

Vague Graph Grammar is an extension of Graph Grammar with added Vague conditions.

Definition 3.31. [81, 190] A *Vague Set (VS) A* on a set X is defined as:

$$A = (t_A, f_A),$$

where:

- $t_A : X \to [0, 1]$ is the *truth-membership function.*
- $f_A : X \to [0, 1]$ is the *falsity-membership function.*

For any element $x \in X$, the following condition holds:

$$0 \leq t_A(x) + f_A(x) \leq 1.$$

Definition 3.32. (cf. [104, 143, 185]) A *Vague Graph (VG)* is defined as a pair:

$$G = (A, B),$$

where:

- $A = (t_A, f_A)$ is a vague set on the vertices V, representing the vertices' truth and falsity degrees.
- $B = (t_B, f_B)$ is a vague set on the edges $E \subseteq V \times V$, representing the edges' truth and falsity degrees.

For each edge $ab \in E$, the following conditions hold:

(1) *Truth Membership Condition:*

$$t_B(ab) \leq \min(t_A(a), t_A(b)).$$

(2) *Falsity Membership Condition:*

$$f_B(ab) \geq \max(f_A(a), f_A(b)).$$

Definition 3.33. Let $G = (A, B)$ and $H = (A', B')$ be two vague graphs. A *Vague Graph Morphism* $f : G \to H$ consists of functions $f_V : V \to V'$ and $f_E : E \to E'$ such that:

- For all $v \in V$:

$$t_A(v) \leq t_{A'}(f_V(v)), \quad f_A(v) \geq f_{A'}(f_V(v)).$$

- For all $e = (u, v) \in E$:

$$f_E(e) = (f_V(u), f_V(v)),$$
$$t_B(e) \leq t_{B'}(f_E(e)),$$
$$f_B(e) \geq f_{B'}(f_E(e)).$$

Definition 3.34. A *Typed Vague Graph* G_T is a triple:

$$G_T = (G, t_G, T),$$

where:

- G is a vague graph.
- T is a vague type graph.
- $t_G : G \to T$ is a total vague graph morphism.

Definition 3.35. A *Vague Rule* p is defined as:

$$p : L \xleftarrow{l} K \xrightarrow{r} R,$$

where:

- L, K, R are typed vague graphs over the vague type graph T.

- l and r are total vague graph morphisms.

Definition 3.36. A *Vague Graph Grammar (VGG)* is a triple:

$$\text{VGG} = (T, G_0, \text{Rule}),$$

where:

- T is the vague type graph.
- G_0 is the initial typed vague graph over T.
- Rule is a set of vague rules.

Theorem 3.37. *A Neutrosophic Graph Grammar (NGG) can be transformed into a Vague Graph Grammar (VGG) by appropriately redefining the neutrosophic membership functions to match the vague membership conditions.*

Proof. Neutrosophic Graph Grammar (NGG) extends conventional graph grammar by incorporating three membership components: truth, indeterminacy, and falsity. To transform an NGG into a VGG, we need to adjust the membership components, reducing the indeterminacy component to zero while retaining the truth and falsity components.

A *Neutrosophic Graph* G is defined as:

$$G = (V, E, \sigma, \mu).$$

The membership degrees satisfy the following conditions for all vertices $v \in V$ and edges $e \in E$:

$$0 \leq \sigma_T(v) + \sigma_I(v) + \sigma_F(v) \leq 3, \quad 0 \leq \mu_T(e) + \mu_I(e) + \mu_F(e) \leq 3.$$

To transform a Neutrosophic Graph Grammar into a Vague Graph Grammar, we redefine the membership functions by eliminating the indeterminacy component and retaining only the truth and falsity components.

We consider Vague Graph Transformation.

(1) *Vertex Transformation:*

For each vertex $v \in V$, we define the vague membership functions as:

$$t_A(v) = \sigma_T(v), \quad f_A(v) = \sigma_F(v).$$

The indeterminacy component is set to zero:

$$\sigma_I(v) = 0.$$

(2) *Edge Transformation:*

For each edge $e \in E$, we define the vague membership functions as:

$$t_B(e) = \mu_T(e), \quad f_B(e) = \mu_F(e).$$

Similarly, the indeterminacy component is set to zero:

$$\mu_I(e) = 0.$$

With these transformations, the vague graph $G = (A, B)$ is formed, where:

- $A = (t_A, f_A)$ represents the vague membership degrees for vertices.
- $B = (t_B, f_B)$ represents the vague membership degrees for edges.

We consider Vague Membership Conditions. The conditions for a Vague Graph are satisfied as follows:

(1) For each vertex $v \in V$:

$$0 \leq t_A(v) + f_A(v) \leq 1,$$

since $\sigma_T(v) + \sigma_F(v) \leq 1$ holds for all vertices in the Neutrosophic Graph.

(2) For each edge $ab \in E$, the vague truth and falsity conditions hold:

$$t_B(ab) \leq \min(t_A(a), t_A(b)),$$
$$f_B(ab) \geq \max(f_A(a), f_A(b)).$$

We consider Vague Graph Grammar Structure. The transformed vague graph G conforms to the structure of a Vague Graph Grammar (VGG):

- The type graph, initial graph, and rules in the NGG are modified by ignoring the indeterminacy component.
- The VGG is defined as:

$$\text{VGG} = (T, G_0, \text{Rule}),$$

where:

  - T is the vague type graph.
  - G_0 is the initial typed vague graph over T.
  - Rule is a set of vague rules.

By setting the indeterminacy components to zero and preserving the truth and falsity components, a Neutrosophic Graph Grammar can be effectively transformed into a Vague Graph Grammar. □

Corollary 3.38. *A Vague Graph Grammar can represent both the empty graph and the null graph.*

Proof. The proof follows almost the same method as for the Fuzzy Graph Grammar. □

3.5. *Plithogenic Graph Grammar*

Plithogenic Graph Grammar is an extension of Graph Grammar with added Plithogenic conditions.

Definition 3.39. [156, 157, 174] A *Plithogenic Graph PG* is defined as:

$$PG = (PM, PN),$$

where:

(1) *Plithogenic Vertex Set* $PM = (M, l, Ml, adf, aCf)$:
- $M \subseteq V$ is the set of vertices.
- l is an attribute associated with the vertices.
- Ml is the set of possible attribute values.
- $adf : M \times Ml \to [0,1]^s$ is the *Degree of Appurtenance Function (DAF)* for vertices.
- $aCf : Ml \times Ml \to [0,1]^t$ is the *Degree of Contradiction Function (DCF)* for vertices.

(2) *Plithogenic Edge Set* $PN = (N, m, Nm, bdf, bCf)$:
- $N \subseteq E$ is the set of edges.
- m is an attribute associated with the edges.
- Nm is the set of possible attribute values.
- $bdf : N \times Nm \to [0,1]^s$ is the *Degree of Appurtenance Function (DAF)* for edges.
- $bCf : Nm \times Nm \to [0,1]^t$ is the *Degree of Contradiction Function (DCF)* for edges.

The Plithogenic Graph PG must satisfy:

(1) *Edge Appurtenance Constraint*:
For all $(x, a), (y, b) \in M \times Ml$:

$$bdf\left((xy), (a, b)\right) \leq \min\{adf(x, a), adf(y, b)\},$$

where $xy \in N$.

(2) *Contradiction Function Constraint*:
For all $(a, b), (c, d) \in Nm \times Nm$:

$$bCf\left((a, b), (c, d)\right) \leq \min\{aCf(a, c), aCf(b, d)\}.$$

(3) *Reflexivity and Symmetry of Contradiction Functions*:

$$\begin{aligned} aCf(a,a) &= 0, & \forall a \in Ml, \\ aCf(a,b) &= aCf(b,a), & \forall a,b \in Ml, \\ bCf(a,a) &= 0, & \forall a \in Nm, \\ bCf(a,b) &= bCf(b,a), & \forall a,b \in Nm. \end{aligned}$$

Definition 3.40. Let $PG = (PM, PN)$ and $PG' = (PM', PN')$ be two plithogenic graphs. A *Plithogenic Graph Morphism* $f : PG \to PG'$ consists of functions:

- $f_V : M \to M'$ mapping vertices.
- $f_E : N \to N'$ mapping edges.

The mappings satisfy:

- Preservation of incidence relations: if $e = xy \in N$, then $f_E(e) = f_V(x)f_V(y) \in N'$.
- Attributes and degrees of appurtenance and contradiction are appropriately mapped and preserved.

Definition 3.41. A *Typed Plithogenic Graph* PG_T is a triple:

$$PG_T = (PG, t_{PG}, TPG),$$

where:

- PG is a plithogenic graph.
- TPG is a plithogenic type graph specifying allowed types and attribute values.
- $t_{PG} : PG \to TPG$ is a total plithogenic graph morphism.

Definition 3.42. A *Plithogenic Rule* p is defined as:

$$p : L \xleftarrow{l} K \xrightarrow{r} R,$$

where:

- L, K, R are typed plithogenic graphs over the plithogenic type graph TPG.
- l and r are total plithogenic graph morphisms.

Definition 3.43. A *Plithogenic Graph Grammar (PGG)* is a triple:

$$\text{PGG} = (TPG, PG_0, \text{Rule}),$$

where:

- TPG is the plithogenic type graph.
- PG_0 is the initial typed plithogenic graph over TPG.
- Rule is a set of plithogenic rules.

Theorem 3.44. *A Plithogenic Graph Grammar (PGG) can be transformed into a Neutrosophic Graph Grammar (NGG), a Vague Graph Grammar (VGG), a Fuzzy Graph Grammar (FGG), and a Turiyam Neutrosophic Graph Grammar (TGG) by appropriately selecting the parameters s and t in the Plithogenic set and redefining its functions accordingly.*

Proof. A Plithogenic Graph Grammar is defined using Plithogenic sets, which generalize various types of fuzzy sets by introducing attributes with possible values and corresponding degrees of appurtenance and contradiction. By specializing the parameters s and t in the Plithogenic set, we can obtain different types of graph grammars.

A *Plithogenic Set PS* is defined as:

$$PS = (P, v, Pv, \mathrm{DAF}, \mathrm{DCF}),$$

.

A *Plithogenic Graph Grammar (PGG)* is defined as:

$$\mathrm{PGG} = (TPG, PG_0, \mathrm{Rule}),$$

where:

- TPG is the plithogenic type graph.
- PG_0 is the initial typed plithogenic graph over TPG.
- Rule is a set of plithogenic rules.

We will show that by selecting appropriate values of s and t and redefining functions, a PGG can be transformed into each of the other graph grammars.

We consider Transformation to a Fuzzy Graph Grammar (FGG). Set $s = t = 1$. The Plithogenic set becomes a Plithogenic Fuzzy Set (PFS). The Degree of Appurtenance Function (DAF) and Degree of Contradiction Function (DCF) reduce to functions with a single value.

- *Vertices and Edges*: The plithogenic vertex set $PM = (M, l, Ml, \mathrm{adf}, \mathrm{aCf})$ reduces to a fuzzy vertex set, where $\mathrm{adf} : M \times Ml \to [0, 1]$ assigns a single membership degree $\mu(v) \in [0, 1]$ to each vertex v.
- The plithogenic edge set $PN = (N, m, Nm, \mathrm{bdf}, \mathrm{bCf})$ reduces similarly for edges.
- *Membership Assignment*: The attributes l and m can be considered constants or omitted since they do not affect the fuzzy membership degrees.
- *Rules and Morphisms*: The rules and morphisms simplify accordingly, with membership degrees represented by single values.

Thus, the PGG reduces to a Fuzzy Graph Grammar (FGG), with the fuzzy membership degrees derived from the degrees of appurtenance in the Plithogenic set.

We consider Transformation to a Neutrosophic Graph Grammar (NGG). Set $s = 3$, $t = 1$. The Plithogenic set becomes a Plithogenic Neutrosophic Set (PNS).

- *Vertices and Edges*: The DAF assigns a vector $(\mu_T(v), \mu_I(v), \mu_F(v)) \in [0,1]^3$ to each vertex v, representing truth, indeterminacy, and falsity membership degrees.
- *Edges*: Similar assignments are made for edges.
- *Membership Constraints*: The degrees satisfy $0 \leq \mu_T(v) + \mu_I(v) + \mu_F(v) \leq 3$, which aligns with the neutrosophic membership conditions.
- *Rules and Morphisms*: The rules and morphisms are adjusted to handle the neutrosophic components, mapping vectors of degrees instead of single values.

Thus, the PGG reduces to a Neutrosophic Graph Grammar (NGG), with the neutrosophic membership degrees derived from the degrees of appurtenance in the Plithogenic set.

We consider Transformation to a Vague Graph Grammar (VGG). Set $s = 2$, $t = 1$, and impose the constraint that for each element x:

$$0 \leq \mu_T(x) + \mu_F(x) \leq 1.$$

This corresponds to a Vague Set.

- *Vertices and Edges*: The DAF assigns a pair $(\mu_T(v), \mu_F(v)) \in [0,1]^2$ to each vertex v, representing truth and falsity membership degrees.
- *Membership Constraints*: The sum of the degrees does not exceed 1, matching the conditions of a Vague Set.
- *Edges*: Similar assignments and constraints apply to edges.
- *Rules and Morphisms*: Adjusted to handle the pair of membership degrees.

Thus, the PGG reduces to a Vague Graph Grammar (VGG), with the vague membership degrees obtained from the degrees of appurtenance in the Plithogenic set.

We consider Transformation to a Turiyam Neutrosophic Graph Grammar (TGG). Set $s = 4$, $t = 1$. The Plithogenic set becomes a Plithogenic Turiyam Neutrosophic Set (PTuS).

- *Vertices and Edges*: The DAF assigns a vector $(\mu_T(v), \mu_I(v), \mu_F(v), \mu_L(v)) \in [0,1]^4$ to each vertex v, representing truth, indeterminacy, falsity, and liberal state membership degrees.
- *Membership Constraints*: The degrees satisfy $0 \leq \mu_T(v) + \mu_I(v) + \mu_F(v) + \mu_L(v) \leq 4$.
- *Edges*: Similar assignments and constraints apply to edges.
- *Rules and Morphisms*: Adjusted to handle the four-component membership degrees.

Thus, the PGG reduces to a Turiyam Neutrosophic Graph Grammar (TGG), with the Turiyam Neutrosophic membership degrees derived from the degrees of appurtenance in the Plithogenic set.

By selecting appropriate values for s and t in the Plithogenic set and redefining the functions accordingly, a Plithogenic Graph Grammar can be transformed into a Neutrosophic Graph

Grammar, a Vague Graph Grammar, a Fuzzy Graph Grammar, or a Turiyam Neutrosophic Graph Grammar. □

Corollary 3.45. *A Plithogenic Graph Grammar can represent both the empty graph and the null graph.*

Proof. The proof follows almost the same method as for the Fuzzy Graph Grammar. □

4. Future prospects

Future prospects are described below.

4.1. *Soft Graph Grammar and Uncertain Hypergraph grammar*

In the future, we plan to explore Soft Graph Grammar. A Soft Graph is a graph that integrates the conditions of a Soft set, while Soft Graph Grammar extends this concept to graph grammar. Numerous studies, including various extensions, have been conducted on Soft sets [38, 55, 59, 66, 68, 89, 155, 166, 187]. Although still in the conceptual stage, the definition is provided below.

Definition 4.1. [17, 108, 114] A *soft set* (F, C) over a universe U is a parameterized family of subsets of U. It is defined as a mapping:

$$F : C \to P(U),$$

where C is a non-empty subset of parameters E, and $P(U)$ denotes the power set of U. For each parameter $c \in C$, $F(c) \subseteq U$ is called the set of c-approximate elements of the soft set (F, C).

Definition 4.2. (cf. [86, 138]) A *soft graph* is defined as a 4-tuple $G = (G', S, T, A)$, where:

(1) $G' = (V, E)$ is a simple graph with vertex set V and edge set E.
(2) A is a non-empty set of parameters.
(3) (S, A) is a soft set over V, where $S : A \to P(V)$.
(4) (T, A) is a soft set over E, where $T : A \to P(E)$.
(5) For each parameter $a \in A$, the pair $F(a) = (S(a), T(a))$ forms a subgraph of G'.

A soft graph can also be represented as:

$$G = (G', S, T, A) = \{F(a) : a \in A\}.$$

We now define soft graph morphisms, which are mappings between soft graphs that preserve the soft structure.

Definition 4.3. Let $G = (G', S, T, A)$ and $H = (H', S', T', B)$ be two soft graphs. A *soft graph morphism* $f : G \to H$ consists of:

(1) A graph homomorphism $f_G : G' \to H'$, which consists of mappings:
- $f_V : V \to V'$ between the vertex sets.
- $f_E : E \to E'$ between the edge sets, satisfying $f_E(e) = (f_V(u), f_V(v))$ for all $e = (u, v) \in E$.

(2) A function $f_A : A \to B$ between the parameter sets.

Such that for all $a \in A$:

(1) $f_V(S(a)) \subseteq S'(f_A(a))$.
(2) $f_E(T(a)) \subseteq T'(f_A(a))$.

Definition 4.4. A *typed soft graph* is a triple $G_T = (G, t_G, T)$, where:

(1) $G = (G', S, T, A)$ is a soft graph.
(2) $T = (G'_T, S_T, T_T, A_T)$ is a soft graph called the *soft type graph.*
(3) $t_G : G \to T$ is a total soft graph morphism, meaning that t_G maps G onto T while preserving the soft graph structure.

Definition 4.5 (Soft Graph Grammar)**.** A *Soft Graph Grammar (SGG)* is a triple:

$$\text{SGG} = (T, G_0, \text{Rule}),$$

where:

- T: Soft type graph specifying permissible labels and parameters.
- G_0: Initial soft graph typed over T.
- Rule: A set of soft rules, where each soft rule p is:

$$p : L \xleftarrow{l} K \xrightarrow{r} R,$$

with L, K, R typed soft graphs over the same soft type graph T, and l, r being total soft graph morphisms.

Theorem 4.6. *Soft Graph Grammar generalizes Graph Grammar.*

Proof. Let $\text{GG} = (T, G_0, \text{Rule})$ be a Graph Grammar with:

- T: Type graph.
- G_0: Initial graph.
- Rule: Set of production rules, $p : G_L \xleftarrow{l} H \xrightarrow{r} G_R$.

To show that GG can be represented as an SGG, define:

- $G' = (V, E)$: Underlying graph structure of G_0.
- $A = \{a\}$: Single parameter.

- $S(a) = V$: Vertex set of G_0 associated with the parameter a.
- $T(a) = E$: Edge set of G_0 associated with the parameter a.
- $F(a) = (S(a), T(a)) = (V, E)$: Subgraph corresponding to the parameter a.

Thus, G_0 is represented as a soft graph $G = (G', S, T, A)$. Similarly, each production rule $p : G_L \xleftarrow{l} H \xrightarrow{r} G_R$ is represented as:

$$p : L \xleftarrow{l} K \xrightarrow{r} R,$$

where L, K, R are soft graphs with a single parameter a, and l, r are mappings preserving the soft structure.

Since every component of GG can be translated into a corresponding component of an SGG, it follows that GG is a special case of SGG with a single parameter. Therefore, SGG generalizes GG. □

Corollary 4.7. *A Soft Graph Grammar can represent both the empty graph and the null graph.*

Proof. The proof follows almost the same method as for the Fuzzy Graph Grammar. □

Furthermore, in the future, we aim to explore Uncertain Hypergraph Grammar, which extends the Uncertain Graph Grammar presented in this paper to hypergraphs (cf. [36, 76, 77, 88, 105, 136, 173]). In the future, we also intend to study Graph Grammar and Automata in the context of Superhypergraphs (cf. [62, 67, 74, 159, 164, 167]).

4.2. *Fuzzy off/over/under automata*

In relation to Fuzzy Sets, the concepts of Fuzzy offset, Fuzzy overset, and Fuzzy underset have been recently introduced [154]. Inspired by these notions, we aim to extend these ideas to the domain of Fuzzy Automata and explore their implications. Although still in the conceptual phase, the definitions are presented as follows.

Definition 4.8 (Fuzzy Overset)**.** (cf. [154]) Let X be a universe of discourse. A *Fuzzy Overset* $\tilde{A}$ in X is defined as:

$$\tilde{A} = \{(x, \mu_{\tilde{A}}(x)) \mid x \in X,\ \mu_{\tilde{A}}(x) \in [0, \Omega]\},$$

where $\Omega > 1$ represents the *Overlimit*, allowing membership degrees greater than 1. There exists at least one $x \in X$ such that $\mu_{\tilde{A}}(x) > 1$.

Definition 4.9 (Fuzzy Underset)**.** (cf. [154]) Let X be a universe of discourse. A *Fuzzy Underset* $\tilde{A}$ in X is defined as:

$$\tilde{A} = \{(x, \mu_{\tilde{A}}(x)) \mid x \in X,\ \mu_{\tilde{A}}(x) \in [\Psi, 1]\},$$

where $\Psi < 0$ is the *Underlimit*, allowing membership degrees below 0. There exists at least one $x \in X$ such that $\mu_{\tilde{A}}(x) < 0$.

Definition 4.10 (Fuzzy Offset)**.** (cf. [154]) Let X be a universe of discourse. A *Fuzzy Offset* $\tilde{A}$ in X is defined as:

$$\tilde{A} = \{(x, \mu_{\tilde{A}}(x)) \mid x \in X,\ \mu_{\tilde{A}}(x) \in [\Psi, \Omega]\},$$

where $\Omega > 1$ and $\Psi < 0$. There exist elements $x, y \in X$ such that $\mu_{\tilde{A}}(x) > 1$ and $\mu_{\tilde{A}}(y) < 0$.

Definition 4.11 (Fuzzy Over General Finite Automata)**.** A *Fuzzy Over General Finite Automaton (FO-GFA)* is an eight-tuple machine:

$$\tilde{F}_{\text{over}} = (Q, \Sigma, \tilde{R}, Z, \tilde{\delta}, \omega, F_1, F_2),$$

where all components are defined as in a General Fuzzy Automaton (GFA), except:

- The membership functions of states and transitions, $\mu(q)$ and $\tilde{\delta}$, map to $[0, \Omega]$, where $\Omega > 1$.
- There exists at least one state $q \in Q$ or transition (q_i, a, q_j) such that $\mu(q) > 1$ or $\tilde{\delta} > 1$.

Definition 4.12 (Fuzzy Under General Finite Automata)**.** A *Fuzzy Under General Finite Automaton (FU-GFA)* is an eight-tuple machine:

$$\tilde{F}_{\text{under}} = (Q, \Sigma, \tilde{R}, Z, \tilde{\delta}, \omega, F_1, F_2),$$

where all components are defined as in a General Fuzzy Automaton (GFA), except:

- The membership functions of states and transitions, $\mu(q)$ and $\tilde{\delta}$, map to $[\Psi, 1]$, where $\Psi < 0$.
- There exists at least one state $q \in Q$ or transition (q_i, a, q_j) such that $\mu(q) < 0$ or $\tilde{\delta} < 0$.

Definition 4.13 (Fuzzy Off General Finite Automata)**.** A *Fuzzy Off General Finite Automaton (FOFF-GFA)* is an eight-tuple machine:

$$\tilde{F}_{\text{off}} = (Q, \Sigma, \tilde{R}, Z, \tilde{\delta}, \omega, F_1, F_2),$$

where all components are defined as in a General Fuzzy Automaton (GFA), except:

- The membership functions of states and transitions, $\mu(q)$ and $\tilde{\delta}$, map to $[\Psi, \Omega]$, where $\Psi < 0$ and $\Omega > 1$.
- There exists at least one state $q \in Q$ or transition (q_i, a, q_j) such that $\mu(q) > 1$, $\mu(q) < 0$, $\tilde{\delta} > 1$, or $\tilde{\delta} < 0$.

Theorem 4.14. *A Fuzzy Off General Finite Automaton (FOFF-GFA) can be transformed into a standard Fuzzy General Finite Automaton (GFA) by restricting its membership degrees to the interval* $[0, 1]$.

Proof. Let $\tilde{F}_{\text{off}} = (Q, \Sigma, \tilde{R}_{\text{off}}, Z, \tilde{\delta}_{\text{off}}, \omega, F_1, F_2)$ be a FOFF-GFA. We construct a corresponding GFA $\tilde{F} = (Q, \Sigma, \tilde{R}, Z, \tilde{\delta}, \omega, F_1, F_2)$ through the following steps:

Each state $q \in Q$ in $\tilde{R}_{\text{off}}$ has a membership degree $\mu_{\text{off}}(q) \in [\Psi, \Omega]$, where $\Psi < 0$ and $\Omega > 1$. Transform $\mu_{\text{off}}(q)$ to $\mu(q) \in [0, 1]$ as follows:

$$\mu(q) = \min(\max(\mu_{\text{off}}(q), 0), 1).$$

The transformed values define the new set of fuzzy start states $\tilde{R}$.

For each transition in $\tilde{\delta}_{\text{off}}$, the membership value $\tilde{\delta}_{\text{off}}((q_i, \mu(q_i)), a, q_j) \in [\Psi, \Omega]$ is restricted to $[0, 1]$ as:

$$\tilde{\delta}((q_i, \mu(q_i)), a, q_j) = \min(\max(\tilde{\delta}_{\text{off}}((q_i, \mu(q_i)), a, q_j), 0), 1).$$

This restriction ensures all transition membership values fall within the interval $[0, 1]$.

The input alphabet Σ, output symbols Z, output function ω, and the membership assignment F_1 and resolution F_2 functions remain unchanged, as they are already defined independently of the interval boundaries.

The resulting automaton $\tilde{F}$ satisfies the definition of a standard GFA because all membership values—both for states and transitions—are now confined to $[0, 1]$. This ensures the transformed automaton adheres to the constraints of a GFA.

Therefore, any FOFF-GFA $\tilde{F}_{\text{off}}$ can be equivalently represented as a GFA $\tilde{F}$ by applying the above restrictions. □

Theorem 4.15. *A Fuzzy Off General Finite Automaton generalizes both the Fuzzy Over General Finite Automaton and the Fuzzy Under General Finite Automaton.*

Proof. This is evident. □

Theorem 4.16. *Every Finite Automaton can be represented as a Fuzzy Off Automaton. Therefore, Fuzzy Off Automata generalize Finite Automata.*

Proof. Let $A = (Q, \Sigma, \delta, q_0, F)$ be a Finite Automaton, where $\delta : Q \times \Sigma \to Q$.

We define a Fuzzy Off Automaton $\tilde{A}_{\text{off}} = (Q, \Sigma, \tilde{\delta}_{\text{off}}, q_0, F)$ with the transition function:

$$\tilde{\delta}_{\text{off}}(q, a, q') = \begin{cases} 1, & \text{if } \delta(q, a) = q', \\ 0, & \text{otherwise.} \end{cases}$$

Here, the membership degrees are within $\{0, 1\}$, which is a subset of $[\Psi, \Omega]$ with $\Psi < 0$ and $\Omega > 1$. This FOA behaves identically to the original FA, with crisp transitions represented by membership degrees of 1 (existing transitions) or 0 (non-existing transitions).

Therefore, every FA can be viewed as a special case of an FOA, proving that FOA generalize FA. □

4.3. *Single-Valued Neutrosophic off automata*

The definition of the Single-Valued Neutrosophic OffSet is also included below [20,31,65,148, 153, 161–163]. We intend to further explore the concept by extending it to automata, specifically by developing and studying the Single-Valued Neutrosophic Off General Neutrosophic Automaton. Additionally, we aim to apply the principles of the Single-Valued Neutrosophic OffSet to Graph Grammar, investigating its potential applications and implications.

Definition 4.17 (Single-Valued Neutrosophic OffSet)**.** [163] A *Single-Valued Neutrosophic OffSet*, denoted $A_{\text{off}} \subseteq U_{\text{off}}$, is a set within a universe of discourse U_{off} in which certain elements may possess neutrosophic degrees—truth, indeterminacy, or falsity—that extend beyond the standard limits, either above 1 or below 0. It is formally defined as:

$$A_{\text{off}} = \{(x, \langle T(x), I(x), F(x)\rangle) \mid x \in U_{\text{off}}, \exists\ (T(x) > 1 \text{ or } F(x) < 0)\},$$

where:

- $T(x)$, $I(x)$, and $F(x)$ denote the truth-membership, indeterminacy-membership, and falsity-membership degrees of each $x \in U_{\text{off}}$.
- $T(x), I(x), F(x) \in [\Psi, \Omega]$, where $\Omega > 1$ (termed the *OverLimit*) and $\Psi < 0$ (termed the *UnderLimit*), allow the possibility for $T(x)$, $I(x)$, or $F(x)$ to take values beyond the conventional bounds of $[0, 1]$.

Definition 4.18 (Single-Valued Neutrosophic Off General Neutrosophic Automaton)**.** A *Single-Valued Neutrosophic Off General Neutrosophic Automaton (SVNO-GNA)* is defined as an eight-tuple machine:

$$\tilde{F}_{\text{off}} = (Q, \Sigma, \tilde{R}_{\text{off}}, Z, \tilde{\delta}_{\text{off}}, \omega, F_1, F_2),$$

where:

- Q, Σ, Z, and ω are defined as in the General Neutrosophic Automaton (GNA).
- $\tilde{R}_{\text{off}} = \{(q, T_0(q), I_0(q), F_0(q)) \mid q \in R\}$ is the set of *off neutrosophic start states*, where:
  - $T_0(q)$ is the truth-membership degree of state q,
  - $I_0(q)$ is the indeterminacy-membership degree of state q,
  - $F_0(q)$ is the falsity-membership degree of state q,

and at least one of $T_0(q) > 1$ or $F_0(q) < 0$.

- $\tilde{\delta}_{\text{off}} : (Q \times [\Psi, \Omega]^3) \times \Sigma \times Q \to [\Psi, \Omega]^3$ is the *neutrosophic off augmented transition function*, where:
  - $[\Psi, \Omega]$ allows membership degrees beyond conventional bounds, with $\Psi < 0$ and $\Omega > 1$.
  - $T(x) > 1$, $F(x) < 0$, or both are possible for states or transitions.
- F_1 and F_2 are the membership assignment and resolution functions, adapted to handle $[\Psi, \Omega]$ intervals.

Theorem 4.19. *A Single-Valued Neutrosophic Off General Neutrosophic Automaton (SVNO-GNA) can be transformed into a standard Neutrosophic General Neutrosophic Automaton (GNA) by restricting its membership degrees within the interval* $[0, 1]$.

Proof. Let $\tilde{F}_{\text{off}} = (Q, \Sigma, \tilde{R}_{\text{off}}, Z, \tilde{\delta}_{\text{off}}, \omega, F_1, F_2)$ be an SVNO-GNA. We construct a corresponding GNA $\tilde{F} = (Q, \Sigma, \tilde{R}, Z, \tilde{\delta}, \omega, F_1, F_2)$ by the following steps:

For each state $q \in Q$, let the truth-membership, indeterminacy-membership, and falsity-membership degrees in $\tilde{R}_{\text{off}}$ be:

$$T_0(q), I_0(q), F_0(q) \in [\Psi, \Omega],$$

where $\Psi < 0$ and $\Omega > 1$. Restrict these values to the interval $[0, 1]$ as follows:

$$T(q) = \min(\max(T_0(q), 0), 1),$$

$$I(q) = \min(\max(I_0(q), 0), 1),$$

$$F(q) = \min(\max(F_0(q), 0), 1).$$

The modified membership degrees $T(q), I(q), F(q)$ define the set of neutrosophic start states $\tilde{R}$.

For each transition $\tilde{\delta}_{\text{off}}((q_i, \mu(q_i)), a, q_j) = (\delta_T, \delta_I, \delta_F)$, where:

$$\delta_T, \delta_I, \delta_F \in [\Psi, \Omega],$$

restrict the values to $[0, 1]$ as follows:

$$\delta'_T = \min(\max(\delta_T, 0), 1),$$

$$\delta'_I = \min(\max(\delta_I, 0), 1),$$

$$\delta'_F = \min(\max(\delta_F, 0), 1).$$

Define the restricted transition function $\tilde{\delta}$ using these values.

The output function ω and the membership assignment and resolution functions F_1, F_2 remain unchanged, as they operate within the specified intervals.

By restricting all membership degrees and transition values to $[0, 1]$, $\tilde{F}_{\text{off}}$ is transformed into a standard GNA $\tilde{F}$ without altering its structural or functional properties. Therefore, the transformation is valid. □

Theorem 4.20. *Every Fuzzy Off Automaton can be represented as a Neutrosophic Off Automaton. Therefore, Neutrosophic Off Automata generalize Fuzzy Off Automata.*

Proof. Let $\tilde{A}_{\text{off}} = (Q, \Sigma, \tilde{\delta}_{\text{off}}, q_0, F)$ be a Fuzzy Off Automaton, where $\tilde{\delta}_{\text{off}} : Q \times \Sigma \times Q \to [\Psi, \Omega]$.

We construct a Neutrosophic Off Automaton $\tilde{A}_{\text{NO}} = (Q, \Sigma, \tilde{\delta}_{\text{NO}}, q_0, F)$ by defining the neutrosophic transition function $\tilde{\delta}_{\text{NO}}$ as:

$$\tilde{\delta}_{\text{NO}}(q, a, q') = (\mu_T(q, a, q'), \mu_I(q, a, q'), \mu_F(q, a, q')),$$

where:

$$\mu_T(q, a, q') = \tilde{\delta}_{\text{off}}(q, a, q'), \quad \mu_I(q, a, q') = 0, \quad \mu_F(q, a, q') = 0.$$

Since $\tilde{\delta}_{\text{off}}$ maps to $[\Psi, \Omega]$, the truth-membership degree $\mu_T(q, a, q')$ also maps to $[\Psi, \Omega]$. The indeterminacy and falsity membership degrees are set to zero for all transitions.

Under this construction, the NOA $\tilde{A}_{\text{NO}}$ captures the behavior of $\tilde{A}_{\text{off}}$, with the fuzziness represented in the truth-membership degrees. Thus, every FOA can be represented as a NOA, demonstrating that NOA generalize FOA. □

Similarly, the Single-Valued Neutrosophic Over General Neutrosophic Automaton and the Single-Valued Neutrosophic Under General Neutrosophic Automaton can also be defined.

4.4. *Finite Hyperautomaton*

In recent years, the concept of a Finite Hyperautomaton has been introduced, known as a generalization of the Finite Automaton. By applying the principles of Superhypergraphs [65, 158, 159], it is anticipated that this framework can be extended to define a Finite SuperhyperAutomaton, and their relationships can be explored. Additionally, there is potential for defining structures such as General Fuzzy Hyperautomaton and General Neutrosophic Hyperautomaton, which merit further investigation. Below, we provide definitions and theorems, including conceptual-level formulations.

Definition 4.21 (Finite Hyperautomaton)**.** (cf. [25, 26]) A *Finite Hyperautomaton (FH)* is a tuple

$$H = (\Sigma, X, Q, Q_0, F, \delta, \alpha),$$

where:

- Σ is a finite alphabet of symbols.

- $X = \{x_1, x_2, \ldots, x_k\}$ is a finite set of word variables.
- Q is a finite set of states.
- $Q_0 \subseteq Q$ is the set of initial states.
- $F \subseteq Q$ is the set of accepting states.
- $\delta \subseteq Q \times (\Sigma \cup \{\#\})^k \times Q$ is the transition relation, where $(\Sigma \cup \{\#\})^k$ is the set of k-tuples formed by elements of Σ padded with a special symbol $\#$.
- α is a quantification condition over the variables in X, expressed as a sequence:

$$\alpha = Q_1 x_1 Q_2 x_2 \ldots Q_k x_k,$$

  where $Q_i \in \{\forall, \exists\}$ indicates whether x_i is universally or existentially quantified.

Remark 4.22 (Key Components and Definitions)**.** (cf. [25, 26])

(1) *Hyperwords:* A *hyperword* over Σ is a finite set of finite words over Σ. A *hyperlanguage* is a set of hyperwords.

(2) *Zipping Function:* Let $s = (w_1, w_2, \ldots, w_k)$ be a tuple of words in a hyperword, where w_i is a finite word over Σ. The *zipping function* $\mathrm{zip}(s)$ produces a word over $(\Sigma \cup \{\#\})^k$, defined as:

$$\mathrm{zip}(s) = s_1 s_2 \ldots s_{\lceil s \rceil},$$

where $s_i[j] = w_j[i]$ for $i \leq |w_j|$, and $s_i[j] = \#$ otherwise. Here, $\#$ is the padding symbol, and $\lceil s \rceil$ is the length of the longest word in s.

(3) *Unzipping Function:* The *unzipping function* reverses the zipping process, mapping a word over $(\Sigma \cup \{\#\})^k$ back to a tuple of words.

(4) *Acceptance Condition:* Let S be a hyperword and $v : X \to S$ be an assignment of word variables to words in S. The hyperword S is *accepted* by H if it satisfies the quantification condition α and the underlying automaton $H^\wedge$ accepts $\mathrm{zip}(v)$.

Definition 4.23 (Acceptance Condition)**.** (cf. [25, 26]) Let α be the quantification condition and $H^\wedge$ be the underlying automaton of H. The satisfaction relation $S \models (\alpha, H)$ is defined as follows:

- If $\alpha = \varepsilon$ (empty quantification), then $S \models (\alpha, H)$ if $H^\wedge$ accepts $\mathrm{zip}(v)$ for some assignment v.
- If $\alpha = \exists x_i \alpha'$, then $S \models (\alpha, H)$ if there exists $w \in S$ such that $S \models (\alpha', H)$ for the assignment $v[x_i \to w]$.
- If $\alpha = \forall x_i \alpha'$, then $S \models (\alpha, H)$ if for all $w \in S$, it holds that $S \models (\alpha', H)$ for the assignment $v[x_i \to w]$.

Definition 4.24 (Accepted Hyperlanguage)**.** (cf. [25, 26]) The *hyperlanguage* of H, denoted $L(H)$, is the set of all hyperwords S such that $S \models H$.

Theorem 4.25. *A Finite Hyperautomaton (FH) is a generalization of a Finite Automaton (FA).*

Proof. Let $A_{\mathrm{FA}} = (\Sigma, Q, Q_0, F, \delta)$ be a Finite Automaton, where:

- Σ is a finite alphabet.
- Q is a finite set of states.
- $Q_0 \subseteq Q$ is the set of initial states.
- $F \subseteq Q$ is the set of accepting states.
- $\delta \subseteq Q \times \Sigma \times Q$ is the transition relation.

We construct a Finite Hyperautomaton $H_{\mathrm{FH}} = (\Sigma, X, Q, Q_0, F, \delta', \alpha)$ that simulates A_{FA}, as follows:

- Set $X = \{x\}$, a singleton set containing one word variable.
- The set of states Q, initial states Q_0, accepting states F, and alphabet Σ are the same as in A_{FA}.
- The transition relation δ' is adapted to operate on tuples over $(\Sigma \cup \{\#\})^1$, which is effectively $\Sigma \cup \{\#\}$. Since we have only one word variable, we can set $\delta' = \delta$.
- The quantification condition is $\alpha = \exists x$, indicating an existential quantification over the word variable x.

In this configuration:

- The hyperwords S are sets of words over Σ.
- The hyperword S is accepted by H_{FH} if there exists a word $w \in S$ such that $H^\wedge$ accepts $\mathrm{zip}(w)$.
- Since $\mathrm{zip}(w) = w$, the underlying automaton $H^\wedge$ operates exactly like A_{FA}.

Therefore, H_{FH} accepts a hyperword S if and only if there exists a word $w \in S$ such that A_{FA} accepts w. This demonstrates that every Finite Automaton can be represented as a Finite Hyperautomaton with one word variable and an existential quantification condition.

Thus, the Finite Hyperautomaton generalizes the Finite Automaton. □

Definition 4.26 (Finite Superhyperautomaton)**.** A *Finite Superhyperautomaton (FSH)* is a tuple

$$H = (\Sigma, X, \mathcal{Q}, \mathcal{Q}_0, \mathcal{F}, \delta, \alpha),$$

where:

- Σ is a finite alphabet of symbols.
- $X = \{x_1, x_2, \ldots, x_k\}$ is a finite set of word variables.
- $\mathcal{Q} \subseteq \mathcal{P}(Q) \setminus \{\emptyset\}$ is the set of *superstates*, where Q is a finite universal set of states.
- $\mathcal{Q}_0 \subseteq \mathcal{Q}$ is the set of initial superstates.

- $\mathcal{F} \subseteq \mathcal{Q}$ is the set of accepting superstates.
- $\delta \subseteq \mathcal{Q} \times (\Sigma \cup \{\#\})^k \times \mathcal{Q}$ is the transition relation.
- α is a quantification condition over the variables in X, expressed as:

$$\alpha = Q_1 x_1 \; Q_2 x_2 \; \ldots \; Q_k x_k,$$

where $Q_i \in \{\forall, \exists\}$ indicates universal or existential quantification over variable x_i.

Remark 4.27 (Key Components and Definitions). (1) *Superstates*:

- Each superstate $S \in \mathcal{Q}$ is a non-empty subset of the universal set of states Q.

(2) *Hyperwords*:

- A *hyperword* over Σ is a finite set of finite words over Σ.
- A *hyperlanguage* is a set of hyperwords.

(3) *Zipping Function*:

- For a tuple $s = (w_1, w_2, \ldots, w_k)$ of words in a hyperword, where each w_i is a finite word over Σ, the *zipping function* $\text{zip}(s)$ produces a word over $(\Sigma \cup \{\#\})^k$, defined as:

$$\text{zip}(s) = s_1 s_2 \ldots s_{\lceil s \rceil},$$

where $s_i[j] = w_j[i]$ if $i \leq |w_j|$, and $s_i[j] = \#$ otherwise. Here, $\#$ is a padding symbol, and $\lceil s \rceil$ is the length of the longest word in s.

(4) *Acceptance Condition*:

- Let S be a hyperword and $v : X \to S$ be an assignment of word variables to words in S.
- The hyperword S is *accepted* by H if it satisfies the quantification condition α and there exists a sequence of superstates in $\mathcal{Q}$ corresponding to the computation over $\text{zip}(v)$ according to the transition relation δ, starting from an initial superstate in $\mathcal{Q}_0$ and ending in an accepting superstate in $\mathcal{F}$.

Definition 4.28 (Acceptance Condition). Let α be the quantification condition and H be the finite superhyperautomaton. The satisfaction relation $S \models (\alpha, H)$ is defined recursively as follows:

- If $\alpha = \varepsilon$ (empty quantification), then $S \models (\alpha, H)$ if there exists an assignment $v : X \to S$ and a computation in H over $\text{zip}(v)$ that leads from an initial superstate in $\mathcal{Q}_0$ to an accepting superstate in $\mathcal{F}$.
- If $\alpha = \exists x_i \alpha'$, then $S \models (\alpha, H)$ if there exists $w \in S$ such that $S \models (\alpha', H)$ with the assignment $v[x_i \to w]$.
- If $\alpha = \forall x_i \alpha'$, then $S \models (\alpha, H)$ if for all $w \in S$, it holds that $S \models (\alpha', H)$ with the assignment $v[x_i \to w]$.

Theorem 4.29. *The Finite Superhyperautomaton (FSH) generalizes both the Finite Hyperautomaton (FH) and the Finite Automaton (FA).*

Proof. To show that the FSH generalizes both FH and FA, we demonstrate that FH and FA are special cases of FSH.

1. FSH Generalizes FH. Let $H_{\mathrm{FH}} = (\Sigma, X, Q, Q_0, F, \delta, \alpha)$ be a Finite Hyperautomaton. We construct an FSH $H_{\mathrm{FSH}} = (\Sigma, X, \mathcal{Q}, \mathcal{Q}_0, \mathcal{F}, \delta', \alpha)$ as follows:

- Set the universal set of states Q of H_{FH} as the universal set for H_{FSH}.
- Define the set of superstates $\mathcal{Q}$ as the set of singleton subsets of Q:

$$\mathcal{Q} = \{\{q\} \mid q \in Q\}.$$

- Similarly, define the initial superstates and accepting superstates as singleton subsets:

$$\mathcal{Q}_0 = \{\{q_0\} \mid q_0 \in Q_0\}, \quad \mathcal{F} = \{\{f\} \mid f \in F\}.$$

- Define the transition relation δ' by lifting the transition relation δ of H_{FH} to operate on singleton superstates:

$$\delta' = \big\{(\{q\}, a, \{q'\}) \mid (q, a, q') \in \delta\big\}.$$

Under this construction, the behavior of H_{FH} is preserved in H_{FSH} because the superstates are singleton sets, and transitions correspond directly to those in H_{FH}. Therefore, FH is a special case of FSH when the superstates are singleton subsets of Q.

2. FSH Generalizes FA. Let $A_{\mathrm{FA}} = (Q, \Sigma, \delta, q_0, F)$ be a Finite Automaton. We construct an FSH $H_{\mathrm{FSH}} = (\Sigma, X, \mathcal{Q}, \mathcal{Q}_0, \mathcal{F}, \delta', \alpha)$ as follows:

- Let $X = \{x\}$ be a singleton set of word variables.
- Set the universal set of states Q of A_{FA} as the universal set for H_{FSH}.
- Define the set of superstates $\mathcal{Q}$ as the set of singleton subsets of Q:

$$\mathcal{Q} = \{\{q\} \mid q \in Q\}.$$

- Define the initial superstate and accepting superstates:

$$\mathcal{Q}_0 = \{\{q_0\}\}, \quad \mathcal{F} = \{\{f\} \mid f \in F\}.$$

- Define the transition relation δ' by lifting δ:

$$\delta' = \big\{(\{q\}, a, \{q'\}) \mid (q, a, q') \in \delta\big\}.$$

- Set the quantification condition $\alpha = \exists x$, indicating that we are interested in the existence of an accepting computation over some word.

In this construction, H_{FSH} accepts a hyperword S if there exists a word $w \in S$ such that the automaton A_{FA} accepts w. Since the superstates are singleton sets, the transitions and states correspond directly to those in A_{FA}. Therefore, FA is a special case of FSH with singleton superstates and existential quantification over a single word variable. □

Funding

This research received no external funding.

Acknowledgments

We humbly extend our heartfelt gratitude to everyone who has provided invaluable support, enabling the successful completion of this paper. We also express our sincere appreciation to all readers who have taken the time to engage with this work. Furthermore, we extend our deepest respect and gratitude to the authors of the references cited in this paper. Thank you for your significant contributions.

Data Availability

This paper does not involve any data analysis.

Ethical Approval

This article does not involve any research with human participants or animals.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Disclaimer

This study primarily focuses on theoretical aspects, and its application to practical scenarios has not yet been validated. Future research may involve empirical testing and refinement of the proposed methods. The authors have made every effort to ensure that all references cited in this paper are accurate and appropriately attributed. However, unintentional errors or omissions may occur. The authors bear no legal responsibility for inaccuracies in external sources, and readers are encouraged to verify the information provided in the references independently. Furthermore, the interpretations and opinions expressed in this paper are solely those of the authors and do not necessarily reflect the views of any affiliated institutions.

References

[1] Kh Abolpour and MM Zahedi. Bl-general fuzzy automata and accept behavior. *Journal of Applied Mathematics and Computing*, 38:103–118, 2012.

[2] Kh Abolpour and MM Zahedi. General fuzzy automata based on complete residuated lattice-valued. *Iranian Journal of Fuzzy Systems*, 14(5):103–121, 2017.

[3] Kh Abolpour and Mohammad Mehdi Zahedi. Isomorphism between two bl-general fuzzy automata. *Soft Computing*, 16(4):729–736, 2012.

[4] Kh Abolpour and Mohammad Mehdi Zahedi. Lb-valued general fuzzy automata. *Fuzzy Sets and Systems*, 442:288–308, 2022.

[5] Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman. Time and tape complexity of pushdown automaton languages. *Inf. Control.*, 13:186–206, 1968.

[6] Alfred V. Aho and Jeffrey D. Ullman. Translations on a context free grammar. *Proceedings of the first annual ACM symposium on Theory of computing*, 1969.

[7] Muhammad Akram. Bipolar fuzzy graphs. *Information sciences*, 181(24):5548–5564, 2011.

[8] Muhammad Akram. Interval-valued fuzzy line graphs. *Neural Computing and Applications*, 21:145–150, 2012.

[9] Muhammad Akram, A Nagoor Gani, and A Borumand Saeid. Vague hypergraphs. *Journal of Intelligent & Fuzzy Systems*, 26(2):647–653, 2014.

[10] Muhammad Akram, Sheng-Gang Li, and KP Shum. Antipodal bipolar fuzzy graphs. *Italian Journal of Pure and Applied Mathematics*, 31(56):425–438, 2013.

[11] Muhammad Akram and Anam Luqman. Intuitionistic single-valued neutrosophic hypergraphs. *Opsearch*, 54:799–815, 2017.

[12] Muhammad Akram, Hafsa M Malik, Sundas Shahzadi, and Florentin Smarandache. Neutrosophic soft rough graphs with application. *Axioms*, 7(1):14, 2018.

[13] Muhammad Akram and Saira Nawaz. Fuzzy soft graphs with applications. *Journal of Intelligent & Fuzzy Systems*, 30(6):3619–3632, 2016.

[14] Muhammad Akram and Musavarah Sarwar. Novel multiple criteria decision making methods based on bipolar neutrosophic sets and bipolar neutrosophic graphs. *viXra*, 2017.

[15] Muhammad Akram and Gulfam Shahzadi. *Operations on single-valued neutrosophic graphs*. Infinite Study, 2017.

[16] Khalid Al-Ahmadi, Linda M. See, Alison J. Heppenstall, and J. Hogg. Calibration of a fuzzy cellular automata model of urban dynamics in saudi arabia. *Ecological Complexity*, 6:80–101, 2009.

[17] M Irfan Ali, Feng Feng, Xiaoyan Liu, Won Keun Min, and Muhammad Shabir. On some new operations in soft set theory. *Computers & Mathematics with Applications*, 57(9):1547–1553, 2009.

[18] Rajeev Alur and David L. Dill. A theory of timed automata. *Theor. Comput. Sci.*, 126:183–235, 1994.

[19] Renzo Angles and Claudio Gutierrez. Survey of graph database models. *ACM Computing Surveys (CSUR)*, 40(1):1–39, 2008.

[20] Nabeel Ezzulddin Arif et al. Domination (set and number) in neutrosophic soft over graphs. *Wasit Journal for Pure sciences*, 1(3):26–43, 2022.

[21] Radim Bělohlávek. Determinism and fuzzy automata. *Information Sciences*, 143(1-4):205–209, 2002.

[22] Alex Bertei, Luciana Foss, Simone André da Costa Cavalheiro, and Renata Hax Sander Reiser. A relational approach of fuzzy graph grammars. *2017 13th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)*, pages 1082–1089, 2017.

[23] Heather Betel and Paola Flocchini. On the relationship between boolean and fuzzy cellular automata. *Electronic Notes in Theoretical Computer Science*, 252:5–21, 2009.

[24] Heather Betel and Paola Flocchini. On the asymptotic behaviour of circular fuzzy cellular automata. *J. Cell. Autom.*, 6:25–52, 2011.

[25] Borzoo Bonakdarpour and Sarai Sheinvald. Automata for hyperlanguages. *arXiv preprint arXiv:2002.09877*, 2020.

[26] Borzoo Bonakdarpour and Sarai Sheinvald. Finite-word hyperlanguages. *Information and Computation*, 295:104944, 2023.

[27] Rajab Ali Borzooei and Hossein Rashmanlou. New concepts of vague graphs. *International Journal of Machine Learning and Cybernetics*, 8:1081–1092, 2017.

[28] Said Broumi and Florentin Smarandache. Several similarity measures of neutrosophic sets. *Infinite Study*, 410(1), 2013.

[29] Said Broumi, Mohamed Talea, Assia Bakali, and Florentin Smarandache. Interval valued neutrosophic graphs. *Critical Review, XII*, 2016:5–33, 2016.

[30] Humberto Bustince and P Burillo. Vague sets are intuitionistic fuzzy sets. *Fuzzy sets and systems*, 79(3):403–405, 1996.

[31] Erick González Caballero, Florentin Smarandache, and Maikel Leyva Vázquez. On neutrosophic offuninorms. *Symmetry*, 11(9):1136, 2019.

[32] Shyi-Ming Chen and Jiann-Mean Tan. Handling multicriteria fuzzy decision-making problems based on vague set theory. *Fuzzy sets and systems*, 67(2):163–172, 1994.

[33] Mario Dal Cin. The algebraic theory of automata. 1980.

[34] Miroslav Ćirić, Jelena Ignjatović, Nada Damljanović, and Milan Bašić. Bisimulations for fuzzy automata. *Fuzzy Sets and Systems*, 186(1):100–139, 2012.

[35] Nicola Cotumaccio and Nicola Prezza. On indexing and compressing finite automata. *ArXiv*, abs/2007.07718, 2020.

[36] Bruno Courcelle, Joost Engelfriet, and Grzegorz Rozenberg. Handle-rewriting hypergraph grammars. *Journal of computer and system sciences*, 46(2):218–270, 1993.

[37] Supriya Kumar De, Ranjit Biswas, and Akhil Ranjan Roy. Some operations on intuitionistic fuzzy sets. *Fuzzy sets and Systems*, 114(3):477–484, 2000.

[38] Irfan Deli. Refined neutrosophic sets and refined neutrosophic soft sets: theory and applications. In *Handbook of research on generalized and hybrid set structures and applications for soft computing*, pages 321–343. IGI Global, 2016.

[39] Narsingh Deo. *Graph theory with applications to engineering and computer science*. Courier Dover Publications, 2016.

[40] Reinhard Diestel. Graph theory 3rd ed. *Graduate texts in mathematics*, 173(33):12, 2005.

[41] Reinhard Diestel. *Graph theory*. Springer (print edition); Reinhard Diestel (eBooks), 2024.

[42] Manfred Droste and Paul Gastin. Weighted automata and weighted logics. *Theor. Comput. Sci.*, 380:69–86, 2005.

[43] Manfred Droste, Werner Kuich, and Heiko Vogler. Handbook of weighted automata. 2009.

[44] Didier Dubois and Henri Prade. A review of fuzzy set aggregation connectives. *Information sciences*, 36(1-2):85–121, 1985.

[45] Didier Dubois and Henri Prade. Fuzzy sets and systems: theory and applications. In *Mathematics in Science and Engineering*, 2011.

[46] Andrzej Ehrenfeucht, Michael G. Main, and Grzegorz Rozenberg. Restrictions on nlc graph grammars. *Theor. Comput. Sci.*, 31:211–223, 1984.

[47] Hartmut Ehrig. Tutorial introduction to the algebraic approach of graph grammars. In *Graph-Grammars and Their Application to Computer Science*, 1986.

[48] Hartmut Ehrig, Hans-Jörg Kreowski, Ugo Montanari, and Grzegorz Rozenberg. Handbook of graph grammars and computing by graph transformation: vol. 3: concurrency, parallelism, and distribution. 1999.

[49] Hartmut Ehrig, Michael Pfender, and Hans Jürgen Schneider. Graph-grammars: An algebraic approach. In *Scandinavian Workshop on Algorithm Theory*, 1973.

[50] Nancy El-Hefenawy, Mohamed A Metwally, Zenat M Ahmed, and Ibrahim M El-Henawy. A review on the applications of neutrosophic sets. *Journal of Computational and Theoretical Nanoscience*, 13(1):936–944, 2016.

[51] Joost Engelfriet, George Leih, and Grzegorz Rozenberg. Apex graph grammars. In *Graph-Grammars and Their Application to Computer Science*, 1986.

[52] Joost Engelfriet, George Leih, and Grzegorz Rozenberg. Apex graph grammars and attribute grammars. *Acta Informatica*, 25:537–571, 1988.

[53] Joost Engelfriet and Jan Joris Vereijken. Context-free graph grammars and concatenation of graphs. *Acta Informatica*, 34:773–803, 1997.

[54] Javier Esparza, A. Kucera, and Richard Mayr. Quantitative analysis of probabilistic pushdown automata: expectations and variances. *20th Annual IEEE Symposium on Logic in Computer Science (LICS' 05)*, pages 117–126, 2005.

[55] Azadeh Zahedi Khameneh etc. Multi-attribute decision-making based on soft set theory: a systematic review. *Soft Computing*, 23:6899 – 6920, 2018.

[56] Kit Fine. Vagueness, truth and logic. 1997.

[57] Mariusz Flasinski. Power properties of nlc graph grammars with a polynomial membership problem. *Theor. Comput. Sci.*, 201:189–231, 1998.

[58] Harary Frank. Shortest paths in probabilistic graphs. *operations research*, 17(4):583–599, 1969.

[59] Takaaki Fujita. Note for hypersoft filter and fuzzy hypersoft filter. *Multicriteria Algorithms With Applications*, 5:32–51, 2024.

[60] Takaaki Fujita. Note for neutrosophic incidence and threshold graph. *SciNexuses*, 1:97–125, 2024.

[61] Takaaki Fujita. A review of the hierarchy of plithogenic, neutrosophic, and fuzzy graphs: Survey and applications. *ResearchGate(Preprint)*, 2024.

[62] Takaaki Fujita. Short note of supertree-width and n-superhypertree-width. *Neutrosophic Sets and Systems*, 77:54–78, 2024.

[63] Takaaki Fujita. Survey of intersection graphs, fuzzy graphs and neutrosophic graphs. *ResearchGate*, July 2024.

[64] Takaaki Fujita. Survey of planar and outerplanar graphs in fuzzy and neutrosophic graphs. *ResearchGate*, July 2024.

[65] Takaaki Fujita. Advancing uncertain combinatorics through graphization, hyperization, and uncertainization: Fuzzy, neutrosophic, soft, rough, and beyond. 2025.

[66] Takaaki Fujita. A comprehensive discussion on fuzzy hypersoft expert, superhypersoft, and indetermsoft graphs. *Neutrosophic Sets and Systems*, 77:241–263, 2025.

[67] Takaaki Fujita and Florentin Smarandache. A concise study of some superhypergraph classes. *Neutrosophic Sets and Systems*, 77:548–593, 2024.

[68] Takaaki Fujita and Florentin Smarandache. A short note for hypersoft rough graphs. *HyperSoft Set Methods in Engineering*, 3:1–25, 2024.

[69] GA Ganati, VNS Rao Repalle, MA Ashebo, and M Amini. Turiyam graphs and its applications. *Information Sciences Letters*, 12(6):2423–2434, 2023.

[70] Gamachu Adugna Ganati, VN Srinivasa Rao Repalle, and Mamo Abebe Ashebo. Social network analysis by turiyam graphs. *BMC Research Notes*, 16(1):170, 2023.

[71] Gamachu Adugna Ganati, VN Srinivasa Rao Repalle, and Mamo Abebe Ashebo. Relations in the context of turiyam sets. *BMC Research Notes*, 16(1):49, 2023.

[72] W-L Gau and Daniel J Buehrer. Vague sets. *IEEE transactions on systems, man, and cybernetics*, 23(2):610–614, 1993.

[73] Ioannis Gerakakis, Prodromos Gavriilidis, Nikolaos I. Dourvas, Ioakeim G. Georgoudas, Giuseppe A. Trunfio, and Georgios Ch. Sirakoulis. Accelerating fuzzy cellular automata for modeling crowd dynamics. *J. Comput. Sci.*, 32:125–140, 2019.

[74] Masoud Ghods, Zahra Rostami, and Florentin Smarandache. Introduction to neutrosophic restricted superhypergraphs and neutrosophic restricted superhypertrees and several of their properties. *Neutrosophic Sets and Systems*, 50:480–487, 2022.

[75] Herbert Göttler. Attributed graph grammars for graphics. In *Graph-Grammars and Their Application to Computer Science*, 1982.

[76] Georg Gottlob, Nicola Leone, and Francesco Scarcello. Hypertree decompositions and tractable queries. In *Proceedings of the eighteenth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 21–32, 1999.

[77] Georg Gottlob, Nicola Leone, and Francesco Scarcello. Hypertree decompositions: A survey. In *Mathematical Foundations of Computer Science 2001: 26th International Symposium, MFCS 2001 Mariánské Lázne, Czech Republic, August 27–31, 2001 Proceedings 26*, pages 37–57. Springer, 2001.

[78] Jonathan L Gross, Jay Yellen, and Mark Anderson. *Graph theory and its applications*. Chapman and Hall/CRC, 2018.

[79] Watkins Gs. Transforming graph grammars into fuzzy graph grammars to recognise noisy two-dimensional images. 2016.

[80] Muhammad Gulistan, Naveed Yaqoob, Zunaira Rashid, Florentin Smarandache, and Hafiz Abdul Wahab. A study on neutrosophic cubic graphs with real life applications in industries. *Symmetry*, 10(6):203, 2018.

[81] Dug Hun Hong and Chang-Hwan Choi. Multicriteria fuzzy decision-making problems based on vague set theory. *Fuzzy sets and systems*, 114(1):103–113, 2000.

[82] John E. Hopcroft and Jeffrey D. Ullman. Introduction to automata theory, languages and computation. 1979.

[83] Liangsong Huang, Yu Hu, Yuxia Li, PK Kishore Kumar, Dipak Koley, and Arindam Dey. A study of regular and irregular neutrosophic graphs with real life applications. *Mathematics*, 7(6):551, 2019.

[84] Dirk Janssens and Grzegorz Rozenberg. Graph grammars with node-label controlled rewriting and embedding. In *Graph-Grammars and Their Application to Computer Science*, 1982.

[85] Dirk Janssens and Grzegorz Rozenberg. A survey of nlc grammars. In *Colloquium on Trees in Algebra and Programming*, 1983.

[86] Jinta Jose, Bobin George, and Rajesh K Thumbakara. Soft graphs: A comprehensive survey. *New Mathematics and Natural Computation*, pages 1–52, 2024.

[87] Cengiz Kahraman. *Fuzzy multi-criteria decision making: theory and applications with recent developments*, volume 16. Springer Science & Business Media, 2008.

[88] Hiroshi Kajino. Molecular hypergraph grammar with its application to molecular optimization. In *International Conference on Machine Learning*, pages 3183–3191. PMLR, 2019.

[89] Hüseyin Kamac and Subramanian Petchimuthu. Bipolar n-soft set theory with applications. *Soft Computing*, 24:16727 – 16743, 2020.

[90] Vasantha Kandasamy, K Ilanthenral, and Florentin Smarandache. *Neutrosophic graphs: a new dimension to graph theory*. Infinite Study, 2015.

[91] V. Karthikeyan, , and R. Karuppaiya. Characterizations of submachine of interval neutrosophic automata. *Advances in Mathematics: Scientific Journal*, 2020.

[92] V. Karthikeyan, , and R. Karuppaiya. Retrievability in interval neutrosophic automata. *Advances in Mathematics: Scientific Journal*, 2020.

[93] V. Karthikeyan, , and R. Karuppaiya. Subsystems of interval neutrosophic automata. *Advances in Mathematics: Scientific Journal*, 2020.

[94] V. Karthikeyan. Characterizations of single valued neutrosophic automata using relations. *Malaya Journal of Matematik*, 2021.

[95] V. Karthikeyan. γ - synchronization of single valued neutrosophic automata. 2021.

[96] V. Karthikeyan and R. Karuppaiya. Strong subsystems of interval neutrosophic automata. 2020.

[97] V. Karthikeyan and R. Karuppaiya. Reverse subsystems of interval neutrosophic automata. 2021.

[98] Jacob Kavikumar, Damian Jan, S. P. Tiwari, Said Broumi, and Florentin Smarandache. Composite neutrosophic finite automata. *Neutrosophic Sets and Systems*, 36:282–291, 2020.

[99] Jacob Kavikumar, Deivanayagampillai Nagarajan, Said Broumi, Florentin Smarandache, Malayalan Lathamaheswari, and Nur Ain Ebas. Neutrosophic general finite automata. 2019.

[100] Changwook Kim. On the structure of linear apex nlc graph grammars. *Theor. Comput. Sci.*, 438:28–33, 2012.

[101] Ekkart Kindler and Robert Wagner. Triple graph grammars : Concepts , extensions , implementations , and application scenarios. 2007.

[102] George Kollios, Michalis Potamias, and Evimaria Terzi. Clustering large probabilistic graphs. *IEEE Transactions on Knowledge and Data Engineering*, 25(2):325–336, 2011.

[103] Alexander Königs and Andy Schürr. Tool integration with triple graph grammars - a survey. In *FoVMT*, 2006.

[104] Saeed Kosari, Zehui Shao, Yongsheng Rao, Xinyue Liu, Ruiqi Cai, and Hossein Rashmanlou. Some types of domination in vague graphs with application in medicine. *Journal of Multiple-Valued Logic & Soft Computing*, 41, 2023.

[105] Hans-Jörg Kreowski, Sabine Kuske, and Aaron Lye. Splicing/fusion grammars and their relation to hypergraph grammars. In *Graph Transformation: 11th International Conference, ICGT 2018, Held as Part of STAF 2018, Toulouse, France, June 25–26, 2018, Proceedings 11*, pages 3–19. Springer, 2018.

[106] Edward T. Lee. Fuzzy tree automata and syntactic pattern recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-4:445–449, 1982.

[107] Yongming Li and Qian Wang. The universal fuzzy automaton. *Fuzzy Sets and Systems*, 249:27–48, 2014.

[108] Pradip Kumar Maji, Ranjit Biswas, and A Ranjan Roy. Soft set theory. *Computers & mathematics with applications*, 45(4-5):555–562, 2003.

[109] Silviu Maniu, Reynold Cheng, and Pierre Senellart. An indexing framework for queries on probabilistic graphs. *ACM Transactions on Database Systems (TODS)*, 42(2):1–34, 2017.

[110] Robert McNaughton. The theory of automata, a survey. *Adv. Comput.*, 2:379–421, 1961.

[111] Jaydev Mishra and Sharmistha Ghosh. Uncertain query processing using vague set or fuzzy set: which one is better? *International Journal of Computers Communications & Control*, 9(6):730–740, 2014.

[112] N. Mohanarao and V. Karthikeyan. Cartesian composition of γ - reset single valued neutrosophic automata. 2021.

[113] Mehryar Mohri. Weighted automata algorithms. 2009.

[114] Dmitri A. Molodtsov. Soft set theory-first results. *Computers & Mathematics With Applications*, 37:19–31, 1999.

[115] John N Mordeson and Premchand S Nair. *Fuzzy graphs and fuzzy hypergraphs*, volume 46. Physica, 2012.

[116] Prem Nath. Context-sensitive grammars and linear-bounded automata. *International Journal of Computer Network and Information Security*, 8:61–66, 2016.

[117] Gia Nhu Nguyen, Le Hoang Son, Amira S Ashour, and Nilanjan Dey. A survey of the state-of-the-arts on neutrosophic sets in biomedical diagnoses. *International Journal of Machine Learning and Cybernetics*, 10:1–13, 2019.

[118] Jörg Niere and Albert Zündorf. Reverse engineering with fuzzy layered graph grammars. 2002.

[119] Priyanka Pal, SP Tiwari, and Shailendra Singh. L-fuzzy rough automaton: a mathematical model for natural languages. *International Journal of Machine Learning and Cybernetics*, 12:2091–2107, 2021.

[120] I. N. Parasyuk and Sergiy Yershov. Categorical approach to the construction of fuzzy graph grammars. *Cybernetics and Systems Analysis*, 42:570–581, 2006.

[121] Theodosios Pavlidis. Linear and context-free graph grammars. *J. ACM*, 19:11–22, 1972.

[122] Georgios A Pavlopoulos, Maria Secrier, Charalampos N Moschopoulos, Theodoros G Soldatos, Sophia Kossida, Jan Aerts, Reinhard Schneider, and Pantelis G Bagos. Using graph theory to analyze biological networks. *BioData mining*, 4:1–27, 2011.

[123] David Pfau, Nicholas Bartlett, and Frank D. Wood. Probabilistic deterministic infinite automata. In *Neural Information Processing Systems*, 2010.

[124] Michael O. Rabin and Dana S. Scott. Finite automata and their decision problems. *IBM J. Res. Dev.*, 3:114–125, 1959.

[125] Diana Raffman. Vagueness without paradox. *The Philosophical Review*, 103(1):41–74, 1994.

[126] Hossein Rashmanlou and Rajab Ali Borzooei. Vague graphs with application. *Journal of Intelligent & Fuzzy Systems*, 30(6):3291–3299, 2016.

[127] Azriel Rosenfeld. Fuzzy graphs. In *Fuzzy sets and their applications to cognitive and decision processes*, pages 77–95. Elsevier, 1975.

[128] Grzegorz Rozenberg. Handbook of graph grammars and computing by graph transformations, volume 1: Foundations. 1997.

[129] Maryam Ghaffari Saadat, Reiko Heckel, and Fernando Orejas. Unfolding symbolic attributed graph grammars. *Graph Transformation*, 12150:75 – 90, 2020.

[130] Rıdvan Şahin. An approach to neutrosophic graph theory with applications. *Soft Computing*, 23(2):569–581, 2019.

[131] Jacques Sakarovitch. Elements of automata theory. 2009.

[132] Arto Salomaa and Ian Naismith Sneddon. Theory of automata. 1969.

[133] Sovan Samanta, Madhumangal Pal, Hossein Rashmanlou, and Rajab Ali Borzooei. Vague graphs and strengths. *Journal of Intelligent & Fuzzy Systems*, 30(6):3675–3680, 2016.

[134] Andy Schürr. Specification of graph translators with triple graph grammars. In *International Workshop on Graph-Theoretic Concepts in Computer Science*, 1994.

[135] Andy Schürr and Felix Klar. 15 years of triple graph grammar : research challenges, new contribultions, open problems. 2008.

[136] Sebastian Seifert and Ingrid Fischer. Parsing string generating hypergraph grammars. In *Graph Transformations: Second International Conference, ICGT 2004, Rome, Italy, September 28–October 1, 2004. Proceedings 2*, pages 352–367. Springer, 2004.

[137] AT Shahida and MS Sunitha. On the metric dimension of join of a graph with empty graph (op). *Electronic Notes in Discrete Mathematics*, 63:435–445, 2017.

[138] Sundas Shahzadi, Musavarah Sarwar, and Muhammad Akram. Decision-making approach with fuzzy type-2 soft graphs. *Journal of Mathematics*, 2020(1):8872446, 2020.

[139] M Shamsizadeh, MM Zahedi, and Kh Abolpour. Bisimulation for bl-general fuzzy automata. 2016.

[140] Marzieh Shamsizadeh, Kh. Abolpour, Ehsan Movahednia, and Manuel de La de La Sen. Vector general fuzzy automaton: A refining analyzing. *International Journal of Analysis and Applications*, 2023.

[141] Marzieh Shamsizadeh, Ehsan Movahednia, and Manuel de la Sen. Isomorphism between two vector general fuzzy automata. *Informatica*, 34:617–633, 2023.

[142] Marzieh Shamsizadeh and Mohammad Mehdi Zahedi. Intuitionistic general fuzzy automata. *Soft Computing*, 20:3505–3519, 2016.

[143] Zehui Shao, Saeed Kosari, Yongsheng Rao, Hossein Rashmanlou, and Farshid Mofidnakhaei. New kind of vague graphs with novel application. *Journal of Multiple-Valued Logic & Soft Computing*, 40, 2023.

[144] Prem Kumar Singh. Data with turiyam set for fourth dimension quantum information processing. *Journal of Neutrosophic and Fuzzy Systems*, 1(1):9–23, 2021.

[145] Prem Kumar Singh. Turiyam set a fourth dimension data representation. *Journal of Applied Mathematics and Physics*, 9(7):1821–1828, 2021.

[146] Prem Kumar Singh. Turiyam set and its mathematical distinction from other sets. *Galoitica: Journal of Mathematical Structures and Applications*, 8(1):08–19, 2023.

[147] Prem Kumar Singh, Naveen Surathu, Ghattamaneni Surya Prakash, et al. Turiyam based four way unknown profile characterization on social networks. *Full Length Article*, 10(2):27–7, 2024.

[148] Florentin Smarandache. Neutrosophic overset, neutrosophic underset, and neutrosophic offset. similarly for neutrosophic over-/under-/offlogic, probability, and statisticsneutrosophic, pons editions brussels, 170 pages book, 2016.

[149] Florentin Smarandache. A unifying field in logics: Neutrosophic logic. In *Philosophy*, pages 1–141. American Research Press, 1999.

[150] Florentin Smarandache. *A unifying field in logics: neutrosophic logic. Neutrosophy, neutrosophic set, neutrosophic probability: neutrsophic logic. Neutrosophy, neutrosophic set, neutrosophic probability.* Infinite Study, 2005.

[151] Florentin Smarandache. Neutrosophic physics: More problems, more solutions. 2010.

[152] Florentin Smarandache. n-valued refined neutrosophic logic and its applications to physics. *Infinite study*, 4:143–146, 2013.

[153] Florentin Smarandache. Degrees of membership>1 and<0 of the elements with respect to a neutrosophic offset. *Neutrosophic Sets and Systems*, 12:3–8, 2016.

[154] Florentin Smarandache. *Neutrosophic Overset, Neutrosophic Underset, and Neutrosophic Offset. Similarly for Neutrosophic Over-/Under-/Off-Logic, Probability, and Statistics.* Infinite Study, 2016.

[155] Florentin Smarandache. Extension of soft set to hypersoft set, and then to plithogenic hypersoft set. *Neutrosophic sets and systems*, 22(1):168–170, 2018.

[156] Florentin Smarandache. *Plithogenic set, an extension of crisp, fuzzy, intuitionistic fuzzy, and neutrosophic sets-revisited.* Infinite study, 2018.

[157] Florentin Smarandache. Plithogeny, plithogenic set, logic, probability, and statistics. *arXiv preprint arXiv:1808.03948*, 2018.

[158] Florentin Smarandache. n-superhypergraph and plithogenic n-superhypergraph. *Nidus Idearum*, 7:107–113, 2019.

[159] Florentin Smarandache. *Extension of HyperGraph to n-SuperHyperGraph and to Plithogenic n-SuperHyperGraph, and Extension of HyperAlgebra to n-ary (Classical-/Neutro-/Anti-) HyperAlgebra.* Infinite Study, 2020.

[160] Florentin Smarandache. *NeutroGeometry & AntiGeometry are alternatives and generalizations of the Non-Euclidean Geometries (revisited)*, volume 5. Infinite Study, 2021.

[161] Florentin Smarandache. *Practical Applications of the Independent Neutrosophic Components and of the Neutrosophic Offset Components.* Infinite Study, 2021.

[162] Florentin Smarandache. Interval-valued neutrosophic oversets, neutrosophic undersets, and neutrosophic offsets. *Collected Papers. Volume IX: On Neutrosophic Theory and Its Applications in Algebra*, page 117, 2022.

[163] Florentin Smarandache. Operators on single-valued neutrosophic oversets, neutrosophic undersets, and neutrosophic offsets. *Collected Papers*, 9:112, 2022.

[164] Florentin Smarandache. *The SuperHyperFunction and the Neutrosophic SuperHyperFunction (revisited again)*, volume 3. Infinite Study, 2022.

[165] Florentin Smarandache. Decision making based on valued fuzzy superhypergraphs. 2023.

[166] Florentin Smarandache. *New types of soft sets "hypersoft set, indetermsoft set, indetermhypersoft set, and treesoft set": an improved version.* Infinite Study, 2023.

[167] Florentin Smarandache. *SuperHyperFunction, SuperHyperStructure, Neutrosophic SuperHyperFunction and Neutrosophic SuperHyperStructure: Current understanding and future directions.* Infinite Study, 2023.

[168] Florentin Smarandache. Short introduction to standard and nonstandard neutrosophic set and logic. *Neutrosophic Sets and Systems*, 77:395–404, 2025.

[169] Florentin Smarandache and Said Broumi. *Neutrosophic graph theory and algorithms.* IGI Global, 2019.

[170] Florentin Smarandache, Said Broumi, Prem Kumar Singh, Chun-fang Liu, V Venkateswara Rao, Hai-Long Yang, Ion Patrascu, and Azeddine Elhassouny. Introduction to neutrosophy and neutrosophic environment. In *Neutrosophic Set in Medical Image Analysis*, pages 3–29. Elsevier, 2019.

[171] Florentin Smarandache, Said Broumi, Mohamed Talea, Assia Bakali, and Kishore Kumar. Shortest path problem on single valued neutrosophic graphs. In *International Symposium on Networks, Computers and Communications (ISNCC-2017), Marrakech, Morocco, May 16-18, 2017, www. isncc-conf. org.* ISNCC, 2017.

[172] Florentin Smarandache, WB Kandasamy, and K Ilanthenral. Applications of bimatrices to some fuzzy and neutrosophic models. 2005.

[173] Grzegorz Soliński, Maciej Woźniak, Jakub Ryzner, Albert Mosiałek, and Anna Paszyńska. Hypergraph grammar-based model of adaptive bitmap compression. In *International Conference on Computational Science*, pages 118–131. Springer, 2020.

[174] Fazeelat Sultana, Muhammad Gulistan, Mumtaz Ali, Naveed Yaqoob, Muhammad Khan, Tabasam Rashid, and Tauseef Ahmed. A study of plithogenic graphs: applications in spreading coronavirus disease (covid-19) globally. *Journal of ambient intelligence and humanized computing*, 14(10):13139–13159, 2023.

[175] Guo-Zheng Sun, C. Lee Giles, and Hsing-Hen Chen. The neural network pushdown automaton: Architecture, dynamics and training. In *Summer School on Neural Networks*, 1997.

[176] Eulalia Szmidt. *Distances and similarities in intuitionistic fuzzy sets*, volume 307. Springer, 2014.

[177] Wolfgang Thomas. A short introduction to infinite automata. In *International Conference on Developments in Language Theory*, 2001.

[178] Salvatore La Torre, P. Madhusudan, and Gennaro Parlato. An infinite automaton characterization of double exponential time. In *Annual Conference for Computer Science Logic*, 2008.

[179] Nenad Trinajstic. *Chemical graph theory.* CRC press, 2018.

[180] Leslie G. Valiant. Regularity and related problems for deterministic pushdown automata. *J. ACM*, 22:1–10, 1975.

[181] Pierluigi Della Vigna and Carlo Ghezzi. Context-free graph grammars. *Inf. Control.*, 37:207–233, 1978.

[182] Haibin Wang, Florentin Smarandache, Yanqing Zhang, and Rajshekhar Sunderraman. *Single valued neutrosophic sets.* Infinite study, 2010.

[183] Greg Watkins. The use of fuzzy graph grammars for recognising noisy two-dimensional images. *Proceedings of North American Fuzzy Information Processing*, pages 415–419, 1996.

[184] Timothy Williamson. *Vagueness.* Routledge, 2002.

[185] Rong Xing, Rui Cheng, Jiwen Huang, Qing Li, and Jingmei Zhao. Learning to understand the vague graph for stock prediction with momentum spillovers. *IEEE Transactions on Knowledge and Data Engineering*, 2023.

[186] Swati Yadav, SP Tiwari, Mausam Kumari, and Vijay K Yadav. Generalized rough and fuzzy rough automata for semantic computing. *International Journal of Machine Learning and Cybernetics*, 13(12):4013–4032, 2022.

[187] Adem Yolcu and Taha Yasin Ozturk. Fuzzy hypersoft sets and it's application to decision-making. *Theory and application of hypersoft set*, 50, 2021.

[188] Lotfi A Zadeh. Fuzzy sets. *Information and control*, 8(3):338–353, 1965.

[189] Lotfi A Zadeh. Fuzzy logic, neural networks, and soft computing. In *Fuzzy sets, fuzzy logic, and fuzzy systems: selected papers by Lotfi A Zadeh*, pages 775–782. World Scientific, 1996.

[190] Shunqi Zeng, Zhenzhi Lin, Fushuan Wen, and Gerard Ledwich. A new approach for power system black-start decision-making with vague set theory. *International Journal of Electrical Power & Energy Systems*, 34(1):114–120, 2012.

[191] H-J Zimmermann. Fuzzy set theory and mathematical programming. *Fuzzy sets theory and applications*, pages 99–114, 1986.

[192] Hans-Jürgen Zimmermann. *Fuzzy set theory—and its applications*. Springer Science & Business Media, 2011.

Received: July 12, 2024. Accepted: September 18, 2024

View publication stats